IBM External Submission #24167

Reprint Courtesy of International Business Machines Corporation, © 2015 International Business Machines Corporation

INTERNATIONAL BUSINESS MACHINES CORPORATION (IBM) ARMONK, NEW YORK 10504

PERMISSION TO REPRINT IBM COPYRIGHTED PUBLICATIONS

Each reprint must be accompanied by the following credit line: **"Reprint Courtesy of International Business Machines Corporation, © (year) International Business Machines Corporation".** The credit line normally should appear on the page where the reprint appears, either under the title or as a footnote.

If the foregoing is inconvenient, the credit line may be placed on the face or back of the title page (or front cover, if there is no title page) or in a conveniently viewable manner with suitable reference to the place where the reprint appears.

When multiple IBM materials are reprinted in the same publication, a consolidated credit paragraph may be used on the title page, or in a conveniently viewable manner listing the titles, corresponding copyright notices and references to the points where the reprints appear.

It is the understanding of **International Business Machines Corporation** that the purpose for which its material is being used is accurate and true as stated in the original request.

Permission to quote from, transmit electronically or reprint IBM material is limited to the purpose and quantities originally requested and must not be construed as a blanket license to use the material for other purposes or to reprint other IBM copyrighted material.

IBM reserves the right to withdraw permission to use copyrighted material whenever, in its discretion, it feels that the privilege of using its material is being used in a way detrimental to its interest or the above instructions are not being followed properly to protect its copyright.

No permission is granted to use trademarks of **International Business Machines Corporation** and its affiliates apart from the incidental appearance of such trademarks in the titles, text, and illustrations of the named publications. Any proposed use of trademarks apart from such incidental appearance requires separate approval in writing and ordinarily cannot be given. The use of any IBM trademark should not be of a manner which might cause confusion of origin or appear to endorse non-IBM products.

THIS PERMISSION IS PROVIDED WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

INTERNATIONAL BUSINESS MACHINES CORPORATION

Dated: August 19, 2015

GPI Guide and Reference

IBM OS/2 Toolkit Documentation

How to Use the GPI Guide and Reference

This reference is a detailed technical guide and reference for application programmers. It gives reference information and code examples to enable you to write source code using Graphical Programming Interface functions.

Before you begin to use this information, it would be helpful to understand how you can:

- Expand the Contents to see all available topics
- Obtain additional information for a highlighted word or phrase
- Use action bar choices
- Use the programming information.

How to Use the Contents

When the Contents window first appears, some topics have a plus (+) sign beside them. The plus sign indicates that additional topics are available.

To expand the Contents if you are using a mouse, click on the plus sign. If you are using the keyboard, use the Up or Down Arrow key to highlight the topic, and press the plus (+) key. For example, **Code Pages** has a plus sign beside it. To see additional topics for that heading, click on the plus sign or highlight that topic and press the plus (+) key.

To view a topic, double-click on the topic (or press the Up or Down Arrow key to highlight the topic, and then press the Enter key).

How to Obtain Additional Information

After you select a topic, the information for that topic appears in a window. Highlighted words or phrases indicate that additional information is available. You will notice that certain words and phrases are highlighted in green letters, or in white letters on a black background. These are called hypertext terms. If you are using a mouse, double-click on the highlighted word. If you are using a keyboard, press the Tab key to move to the highlighted word, and then press the Enter key. Additional information then appears in a window.

How to Use Action Bar Choices

Several choices are available for managing information presented in the *Graphics Programming Interface Programming Reference* . There are three pull-down menus on the action bar: the **Services** menu, the **Options** menu, and the **Help** menu.

The actions that are selectable from the **Services** menu operate on the active window currently displayed on the screen. These actions include the following:

Bookmark

Allows you to set a placeholder so you can retrieve information of interest to you.

When you place a bookmark on a topic, it is added to a list of bookmarks you have previously set. You can view the list, and you can remove one or all bookmarks from the list. If you have not set any bookmarks, the list is empty.

To set a bookmark, do the following:

1. Select a topic from the Contents.
2. When that topic appears, choose the **Bookmark** option from the **Services** pull-down.
3. If you want to change the name used for the bookmark, type the new name in the field.
4. Click on the **Place** radio button (or press the Up or Down Arrow key to select it).
5. Click on **OK** (or select it and press Enter). The bookmark is then added to the bookmark list.

Search

Allows you to find occurrences of a word or phrase in the current topic, selected topics, or all topics.

You can specify a word or phrase to be searched. You can also limit the search to a set of topics by first marking the topics in the Contents list.

To search for a word or phrase in all topics, do the following:

1. Choose the **Search** option from the **Services** pull-down.
2. Type the word or words to be searched for.
3. Click on **All sections** (or press the Up or Down Arrow keys to select it).

4. Click on **Search** (or select it and press Enter) to begin the search.

5. The list of topics where the word or phrase appears is displayed.

Print

Allows you to print one or more topics. You can also print a set of topics by first marking the topics in the Contents list.

To print the document Contents list, do the following:

1. Choose **Print** from the **Services** pull-down.

2. Click on **Contents** (or press the Up or Down Arrow key to select it).

3. Click on **Print** (or select it and press Enter).

4. The Contents list is printed on your printer.

Copy

Allows you to copy a topic that you are viewing to the System Clipboard or to a file that you can edit. You will find this particularly useful for copying syntax definitions and program samples into the application that you are developing.

You can copy a topic that you are viewing in two ways:

- **Copy** copies the topic that you are viewing into the System Clipboard. If you are using a Presentation Manager editor (for example, the System Editor) that copies or cuts (or both) to the System Clipboard, and pastes to the System Clipboard, you can easily add the copied information to your program source module.

- **Copy to file** copies the topic that you are viewing into a temporary file named TEXT.TMP. You can later edit that file by using any editor. You will find TEXT.TMP in the directory where your viewable document resides.

 To copy a topic, do the following:

 1. Expand the Contents list and select a topic.

 2. When the topic appears, choose **Copy to file** from the **Services** pull-down.

 3. The system puts the text pertaining to that topic into the temporary file named TEXT.TMP.

For information on one of the other choices in the **Services** pull-down, highlight the choice and press the F1 key.

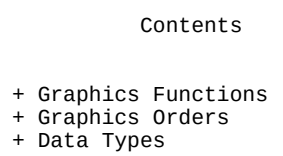
The actions that are selectable from the **Options** menu allow you to change the way your Contents list is displayed. To expand the Contents and show all levels for all topics, choose **Expand all** from the **Options** pull-down. You can also press the Ctrl and * keys together. For information on one of the other choices in the **Options** pull-down, highlight the choice and press the F1 key.

The actions that are selectable from the **Help** menu allow you to select different types of help information. You can also press the F1 key for help information about the Information Presentation Facility (IPF).

How to Use the Programming Information

This document consists of guide and reference information that provides a detailed description of each function, message, constant, and data type. It provides language-dependent information about the functions which enable the user to generate call statements in the C Language.

Graphical Programming Interface programming information is presented by component, such as Graphics Functions, Graphics Orders, and Data Types, for example:



By clicking on the plus sign beside "Graphics Functions", you see an alphabetic list of the Graphical Programming Interface functions. Selecting a function takes you directly into the reference information for that function.

Units of reference information are presented in selectable multiple windows or viewports. A viewport is a Presentation Manager window that can be sized, moved, minimized, maximized, or closed. By selecting a unit (in this case, an entry on the Contents list), you will see two windows displayed:

```
  Unit Title               Selection Title

  Select an item

  Syntax
  Returns
  Notes
  Example Code
  Related Functions
  Glossary
```

The window on the left is the primary window. It contains a list of items that are always available to you. The window on the right is the secondary window. It contains a "snapshot" of the unit information. For reference units (that is, function descriptions), this window contains the Function Syntax.

All of the information needed to understand a reference unit (or topic) is readily available to you through the primary window. The information is divided into discrete information groups, and only the appropriate information group appears for the topic that you are viewing.

The information groups for a reference unit (that is, a function description) can include all or some of the following:

- Syntax
- Parameters
- Returns
- Errors
- Notes
- Example Code
- Related Functions
- Graphic Elements and Orders
- Glossary

This list may vary. Some topics may be omitted when they do not apply.

Information groups are displayed in separate viewports that are stacked in a third window location that overlaps the secondary window. By selecting an item (information group) in the primary window, the item is displayed in the third window location, as follows:

```
Unit Title         Selection     Glossary

Select an item                Select a starting
                              letter of
   .                          glossary terms
   .
   .                          A    N
   .                          B    O
   .                          C    P
Glossary                      .    .
                              .    .
                              .    .
                              M    Z
```

By selecting successive items from the primary window, additional windows are displayed on top of the previous windows displayed in the third window location. For example, in a function description, Parameters and Return Values are items listed in the primary window. When selected, they appear one on top of the other in the third window location. Because of this, you may move the first selected (topmost) window to the left before selecting the next item. This allows simultaneous display of two related pieces of information from the "stack" of windows in the third window location, as follows:

```
Unit Title        Parameters     Return Values

Select an item
   .
   .
   .
Returns
Errors
   .
   .
   .
```

Each window can be individually closed from its system menu. All windows are closed when you close the primary window.

Some secondary windows may have the appearance of a split screen. For example, an illustration may appear in the left half of the window, and scrollable, explanatory information may appear in the right half of the window. Because illustrations may not necessarily fit into the small window size on your screen, you may maximize the secondary window for better readability.

Conventions Used in this Reference

The purpose of this reference is to give information about functions, constants, and data types. It provides information about the functions which enables the user to call functions in the C programming language.

The following information is provided:

- The syntax and parameters for each function.
- The syntax of each data type and structure

Notation Conventions

The following notation conventions are used in this reference:

NULL	The term NULL applied to a parameter is used to indicate the presence of the pointer parameter, but with no value.
NULLHANDLE	The term NULLHANDLE applied to a parameter is used to indicate the presence of the handle parameter, but with no value.
Implicit Pointer	If no entry for a data type "Pxxxxxxx" is found in Data Types, then it is implicitly a pointer to the data type "xxxxxxx". See Implicit Pointer Data Types for more information about implicit pointers.
Constant Names	All constants are written in uppercase to match the header files. Where applicable, constant names have a prefix derived from the name of a function, message, or idea associated with the constant. For example: WM_CREATE — Window message SV_CXICON — System value CF_TEXT — Clipboard format. In this reference, references to a complete set of constants with the same prefix is written as shown in the following examples: Window message — WM_* System value — SV_*
Parameters and Fields	Function parameters and data structure fields are shown in italics.

Conventions Used in Function Descriptions

The documentation of each function contains the following sections:

Syntax
The function syntax describes the C-language calling syntax of the function and gives a brief description.

Programming Note

The functions in this book are spelled in mixed-case for readability but are known to the system as uppercase character strings. For example, the function "GPIBeginArea" is actually the external name "GPIBEGINAREA".

If you are using a compiler that generates a mixed-case external name, you should code the functions in uppercase.

Parameters

Each parameter is listed with its C-language data type, parameter type, and a brief description.

- All data types are written in uppercase to match the header files. A data type of "Pxxxxxxx" implicitly defines a pointer to the data type "xxxxxxx".

 The term NULL applied to a parameter indicates the presence of the parameter, with no value.

 Refer to Data Types for a complete list of all data types and their descriptions.

- There are three parameter types:

Input	Specified by the programmer.
Output	Returned by the function.
Input/Output	Specified by the programmer and modified by the function.

- A brief description is provided with each parameter. Where appropriate, restrictions are also included. In some cases, the parameter points to a structure.

Returns

A list of possible return codes or errors (when appropriate) is included in this section. Some functions do not have return codes. Refer to Error Number and Name for a list of error codes and their numerical values, and Error Name and Explanation for a list of error codes and their descriptions.

Remarks

This section contains additional information about the function, when required.

Related Functions

This list shows the functions (if any) that are related to the function being described.

Example Code

An example of how the function can be used is shown in C language.

Error Severities

Each of the error conditions given in the list of errors for each function falls into one of these areas:

Warning
: The function detected a problem, but took some remedial action that enabled the function to complete successfully. The return code in this case indicates that the function completed successfully.

Error
: The function detected a problem for which it could not take any sensible remedial action. The system has recovered from the problem, and the state of the system, with respect to the application, remains the same as at the time when the function was requested. The system has not even partially executed the function (other than reporting the error).

Severe Error
: The function detected a problem from which the system could not reestablish its state, with respect to the application, at the time when that function was requested; that is, the system partially executed the function. This necessitates the application performing some corrective activity to restore the system to some known state.

Unrecoverable Error
: The function detected some problem from which the system could not re-establish its state, with respect to the application, at the time when that call was issued. It is possible that the application cannot perform some corrective action to restore the system to some known state.

The WinGetLastError and WinGetErrorInfo functions can be used to find out more about an error (or warning) that occurs as a result of

executing a call.

Header Files

All functions require an "#include" statement for the system header file OS2.H:

```
#include  <OS2.H>
```

Most functions also require a "#define" statement to select an appropriate (conditional) section of the header file, and hence, the required prototype. Where this is necessary, it is shown at the head of the function definition in the form:

```
#define   INCL_name
```

Note: These "#define" statements must precede the "#include <OS2.H>" statement.

Addressing Elements in Arrays

Constants defining array elements are given values that are zero-based in C; that is, the numbering of the array elements starts at zero, not one.

For example, in the DevQueryCaps function, the sixth element of the *alArray* parameter is CAPS_HEIGHT, which is equated to 5.

Count parameters related to such arrays always mean the actual number of elements available; therefore, again using the DevQueryCaps function as an example, if all elements up to and including CAPS_HEIGHT are provided for, *lCount* could be set to (CAPS_HEIGHT+1).

In functions for which the starting array element can be specified, this is always zero-based, and so the C element number constants can be used directly. For example, to start with the CAPS_HEIGHT element, the DevQueryCaps parameter can be set to CAPS_HEIGHT.

Implicit Pointer Data Types

A data type name beginning with "P" (for example, PERRORCODE) is likely to be a pointer to another data type (in this instance, ERRORCODE).

In the data type summary, Data Types, no explicit "typedefs" are shown for pointers; therefore, if no data type definition can be found in the summary for a data type name "Pxxxxxx", it represents a pointer to the data type "xxxxxx", for which a definition should be found in the reference.

The implicit type definition needed for such a pointer "Pxxxxxx" is:

```
typedef xxxxxx *Pxxxxxx;
```

Such definitions are provided in the header files. OS2.H.

Storage Mapping of Data Types

The storage mapping of the data types is dependent on the machine architecture. To be portable, applications must access the data types using the definitions supplied for the environment in which they will execute.

Double-Byte Character Set (DBCS)

Throughout this publication, you will see references to specific value for character strings. The values are for single-byte character set (SBCS). If you use the double-byte character set (DBCS), note that one DBCS character equals two SBCS characters.

Programming Considerations

This section provides information you need to consider before you begin programming with GPI functions.

Stack Size

Existing 16-bit applications (small and tiny models) must have a 4KB stack available when they enter system calls; otherwise, the stack can overflow into the data area.

Presentation Manager

The Presentation Manager component of the OS/2* operating system is based on the IBM Systems Application Architecture* (SAA*) Common Programming Interface-a an architecture for the design and development of applications.

The Presentation Manager component implements the Common User Access* (CUA*) interface, which you can use to attain consistency in the appearance and behavior of your applications.

C++ Considerations

This section contains several topics you should take into consideration if you are using C++ **.

C++ Header Files

OS/2 functions that used to take a PSZ as a parameter, and that do not modify the contents of the passed string, have been updated in the C++ header files to take a PCSZ data type parameter. The use of PCSZ allows for better optimization by the compiler and is more semantically compatible with C++. Existing code that calls functions that use PSZ will continue to work correctly.

Several of the typedefs have been changed in the C++ header files. For example, many items that are unsigned char in the C header files are char in the C++ header files. For instance,

```
typedef unsigned char BYTE;
```

has changed to

```
typedef char BYTE;
```

The existing samples that are included in the IBM Developer's Toolkit for OS/2 Warp, Version 3 can be used with either set of the header files.

PCSZ Data Type

Note: The PCSZ data type is defined in the C++ header files included with this product. The use of the "const" keyword is not necessarily specific to C++. Certain C compilers support it as well.

If a function takes as a parameter a string that is not changed by the function, the string parameter can be declared as a "const" string, or a PCSZ. PCSZ is defined in the C++ header files as a "const" pointer to a NULL-delimited string. The "const" means that the function will not change the contents of the string.

Declaring the parameter as PCSZ informs the C++ compiler that the function will not change the string. Therefore, the compiler simply passes a pointer to the string in the function parameter list. If the parameter is declared as a normal PSZ (not "const"), the compiler assumes that the function might change the string. Under these circumstances the compiler will add code to make a copy of the string then pass a pointer to the copy, rather than pass a pointer to the original string.

A smaller, faster executable is often produced if the data item passed in a parameter list is declared as "const".

If the data item is declared as "const" then it must not be changed by the function.

LINK386

The C++ compiler will provide a dynamic link library which is be used by LINK386 when generating error messages. This DLL will convert a compiler generated mangled name into the function prototype. If the DLL is not present, an error message will be displayed and LINK386 will display the compiler-generated mangled name in error messages.

Graphics Functions

Coordinates

GPI coordinate values that are in world or model space are passed in variables of data type LONG. For a presentation space of format GPIF_LONG (see GpiCreatePS), the signed value must be contained within the low-order 28 bits.

For a presentation space with a format of GPIF_SHORT, the signed value must be contained within the low-order 16 bits. Coordinates that exceed this limit are truncated without error, when stored in a segment. As a consequence, a large positive number may appear as a negative number.

In both instances, after transformation to media space (that is, device space, possibly including a translation for the window origin), coordinate values must be in the range -32 768 through +32 767.

The PMERR_COORDINATE_OVERFLOW error condition occurs if a coordinate is too large to be handled.

Region coordinates must be within the range -32 767 through +32 765.

Matrix Parameter Values

These GPI functions define transforms:

- GpiSetSegmentTransformMatrix
- GpiSetModelTransformMatrix
- GpiCallSegmentMatrix
- GpiSetViewingTransformMatrix
- GpiSetDefaultViewMatrix
- GpiCreatePS
- GpiSetPageViewport.

Note: The last two functions define the device transform; the page viewport may be defaulted.

Concatenation of transform matrixes can occur as the transform is specified, for example, if TRANSFORM_ADD is specified. Concatenation also occurs during drawing, between the various transforms in the viewing pipeline.

During the process of concatenation, it is possible for the matrix parameter overflow error, PMERR_INV_MATRIX_ELEMENT, to occur. This error is raised if either of the following conditions occurs for any intermediate value during the concatenation arithmetic (see, for example, GpiSetSegmentTransformMatrix for an explanation of matrix element numbers):

- Any of the matrix elements 1, 2, 4, or 5 is greater than 32 767 or less than -32 768 (±1 for a GPIF_SHORT format presentation space), or
- Either of elements 7 or 8 is greater than 134 217 727 (2^{27} -1) or less than -134 217 728 (-2^{27}) (greater than 32 767 or less than -32 768 for a GPIF_SHORT format presentation space).

Rounding Errors

In general for graphics coordinates, when non-unity transforms (apart from simple translation) are involved, rounding errors occur. For example, adding the coordinates of one point to a delta value, to produce the coordinates of a second point (all in world coordinates) does not always map to the same device pel as if the computation had been done in device coordinates. Such errors can be avoided if calculations are done in device coordinates, or if there are no scaling (or rotational, or shear) elements in the transforms. Alternatively, the problems can be reduced, though not eliminated, by defining very fine world coordinates.

Drawing Process Check Errors

Some GPI functions involve processing buffers of graphics orders or retained graphics segments (the data for which consists of graphics orders). These functions can give rise to Drawing Process Check (DPC) errors if an order is found that either is not valid in its context or that contains invalid data. If this happens, processing of the function stops and the error is recorded. Note that orders up to the one found to be in error are processed by the function, and output occurs if drawing is being performed.

Each function that can return these errors has Drawing Process Check errors in its error condition list. The full list of DPC errors is:

PMERR_INV_IN_AREA
PMERR_INV_IN_PATH
PMERR_INV_IN_ELEMENT
PMERR_ALREADY_IN_ELEMENT
PMERR_STOP_DRAW_OCCURRED (warning)
PMERR_PATH_INCOMPLETE
PMERR_AREA_INCOMPLETE
PMERR_IMAGE_INCOMPLETE

PMERR_INV_ORDER_LENGTH
PMERR_NOT_IN_IMAGE
PMERR_NOT_IN_AREA
PMERR_NOT_IN_ELEMENT
PMERR_NOT_IN_PATH
PMERR_INSUFFICIENT_MEMORY
PMERR_SEG_CALL_STACK_EMPTY
PMERR_SEG_CALL_STACK_FULL
PMERR_TRUNCATED_ORDER
PMERR_CALLED_SEG_NOT_FOUND
PMERR_DYNAMIC_SEG_SEQ_ERROR
PMERR_PROLOG_ERROR
PMERR_INV_IN_VECTOR_SYMBOL

GpiAnimatePalette

GpiAnimatePalette - Syntax

This function changes the color values of animating indexes in a palette.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPAL      hpal;      /*  Palette handle. */
ULONG     ulFormat;  /*  Format of entries in the table. */
ULONG     ulStart;   /*  Starting index. */
ULONG     ulCount;   /*  Count of elements in aulTable. */
PULONG    aulTable;  /*  Start of the application data area. */
LONG      lChanged;  /*  Number of remapped colors. */

lChanged = GpiAnimatePalette(hpal, ulFormat,
             ulStart, ulCount, aulTable);
```

GpiAnimatePalette Parameter - hpal

hpal (HPAL) - input
Palette handle.

GpiAnimatePalette Parameter - ulFormat

ulFormat (ULONG) - input
Format of entries in the table.

It must be the following value:

LCOLF_CONSECRGB
Array of RGB values, corresponding to color indexes *ulStart* upwards. Each entry is 4 bytes long.

GpiAnimatePalette Parameter - ulStart

ulStart (ULONG) - input
Starting index.

This is relevant only for LCOLF_CONSECRGB.

GpiAnimatePalette Parameter - ulCount

ulCount (ULONG) - input
Count of elements in *aulTable* .

This must be greater than or equal to 0.

GpiAnimatePalette Parameter - aulTable

aulTable (PULONG) - input
Start of the application data area.

This contains the palette definition data. The format depends on the value of *ulFormat* .

Each color value is a 4-byte integer, with a value of

```
(F * 16777216) + (R * 65536) + (G * 256) + B
```

where:

F — is a flag byte, which can take the following values (these can be ORed together if required):

PC_RESERVED	This index is an animating index. This means that the application might frequently change the RGB value, so the system should not map the logical index of the palette of another application to the entry in the physical palette used for this color.
PC_EXPLICIT	The low-order word of the logical color table entry designates a physical palette entry. This allows an application to show the contents of the device palette as realized for other logical palettes. This does not prevent the color in the entry from being changed for any reason.

R — is red intensity value

G — is green intensity value

B — is blue intensity value. The maximum intensity for each primary is 255.

GpiAnimatePalette Return Value - lChanged

lChanged (LONG) - returns
Number of remapped colors.

PAL_ERROR
Error occurred

Other
Number of colors remapped (that is, having entries in the physical color table). These are all animating indexes: they have the PC_RESERVED flag set on this function. If the palette is selected into more than one presentation space, the number returned is the maximum number of indexes that have entries in any of the relevant devices.

Note: By the time an application receives this information, other applications using the palette may have caused the number to be changed.

GpiAnimatePalette - Parameters

hpal (HPAL) - input
Palette handle.

ulFormat (ULONG) - input
Format of entries in the table.

It must be the following value:

LCOLF_CONSECRGB
Array of RGB values, corresponding to color indexes *ulStart* upwards. Each entry is 4 bytes long.

ulStart (ULONG) - input
Starting index.

This is relevant only for LCOLF_CONSECRGB.

ulCount (ULONG) - input
Count of elements in *aulTable* .

This must be greater than or equal to 0.

aulTable (PULONG) - input
Start of the application data area.

This contains the palette definition data. The format depends on the value of *ulFormat* .

Each color value is a 4-byte integer, with a value of

```
(F * 16777216) + (R * 65536) + (G * 256) + B
```

where:

F is a flag byte, which can take the following values (these can be ORed together if required):

PC_RESERVED	This index is an animating index. This means that the application might frequently change the RGB value, so the system should not map the logical index of the palette of another application to the entry in the physical palette used for this color.
PC_EXPLICIT	The low-order word of the logical color table entry designates a physical palette entry. This allows an application to show the contents of the device palette as realized for other logical palettes.

This does not prevent the color in the entry from being changed for any reason.

R — is red intensity value

G — is green intensity value

B — is blue intensity value. The maximum intensity for each primary is 255.

lChanged (LONG) - returns
Number of remapped colors.

PAL_ERROR
Error occurred

Other
Number of colors remapped (that is, having entries in the physical color table). These are all animating indexes: they have the PC_RESERVED flag set on this function. If the palette is selected into more than one presentation space, the number returned is the maximum number of indexes that have entries in any of the relevant devices.

Note: By the time an application receives this information, other applications using the palette may have caused the number to be changed.

GpiAnimatePalette - Remarks

The animating indexes are those that have the PC_RESERVED flag set in the palette and also in the corresponding element of the *aulTable* array in this function.

If an animating index already has an entry in the physical hardware palette (allocated from a previous call to WinRealizePalette), both that entry and the entry in the logical palette are changed. If there is not an entry in the physical palette, or the device does not support palette functions, the logical palette color is changed. This function does not allocate a new entry in the physical palette.

This function ignores those elements in *aulTable* corresponding to non-animating indexes (those that do not have the PC_RESERVED flag set). Their colors are not changed.

All presentation spaces that have this palette selected into them (see GpiSelectPalette) are updated with the effects of this function. It is not necessary to issue a WinRealizePalette function before the effects become visible.

If a palette is selected into a presentation space that is associated with a device context of type OD_METAFILE or OD_METAFILE_NOQUERY, only the final color values are recorded in the metafile.

It is an error if a palette is selected into a presentation space that is within an area or path definition when this function is issued.

GpiAnimatePalette - Errors

Possible returns from WinGetLastError

PMERR_INV_HPAL (0x2111)
An invalid color palette handle was specified.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COLOR_DATA (0x2054)
Invalid color table definition data was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_FORMAT (0x2055)
An invalid format parameter was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_START_INDEX (0x2058)
An invalid starting index parameter was specified with a logical color table or color query function.

PMERR_INSUFFICIENT_MEMORY (0x203E)
The operation terminated through insufficient memory.

PMERR_PALETTE_BUSY (0x2112)
An attempt has been made to reset the owner of a palette when it was busy.

PMERR_INV_IN_AREA (0x2085)
An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

GpiAnimatePalette - Related Functions

Related Functions

- GpiCreatePalette
- GpiDeletePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSelectPalette
- GpiSetPaletteEntries

GpiAnimatePalette - Example Code

This example uses GpiAnimatePalette to change the color values of the first four animating indexes in a palette.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

LONG  lremapColors;     /* number of remapped colors            */
HPAL  hpal;             /* palette handle                       */

/*************************************************************
 * assume 4 entries in palette.                               *
 * The RGB values are calculated with the following formula:  *
 *    (F * 16777216) + (R * 65536) + (G * 256) + B            *
 *    where F = flag, PC_RESERVED or PC_EXPLICIT              *
 *          R = red intensity value                           *
 *          G = green intensity value                         *
 *          B = blue intensity value                          *
 * Thus, in the following table, red and green intensities are 0 *
 * while the blue intensity increases from 1 to 4.            *
 *************************************************************/

ULONG aulTable[4]=
          {(PC_RESERVED*16777216) + (0*65536) + (0*256) + 1,
           (PC_RESERVED*16777216) + (0*65536) + (0*256) + 2,
           (PC_RESERVED*16777216) + (0*65536) + (0*256) + 3,
           (PC_RESERVED*16777216) + (0*65536) + (0*256) + 4};


lremapColors = GpiAnimatePalette(hpal, LCOLF_CONSECRGB, 0L, 4L,
                                 aulTable);
```

GpiAnimatePalette - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiAssociate

GpiAssociate - Syntax

This function associates a graphics presentation space with, or dissociates it from, a device context.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
HDC     hdc;  /*  Device-context handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiAssociate(hps, hdc);
```

GpiAssociate Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiAssociate Parameter - hdc

hdc (HDC) - input
Device-context handle.

GpiAssociate Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiAssociate - Parameters

hps (HPS) - input
Presentation-space handle.

hdc (HDC) - input
Device-context handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiAssociate - Remarks

Any type of device context may be used.

Subsequent drawing functions direct output to the associated device context.

If a null handle is supplied for the device context, the presentation space is dissociated from its currently-associated device context. An associated presentation space cannot be associated with another device context, and an associated device context cannot be associated with another presentation space.

An error occurs if you try to draw to a presentation space associated with a memory device context that has no bit map selected into it (see GpiSetBitmap).

The processing described for GRES_ATTRS (see GpiResetPS) is performed on the presentation space. Also, bounds data is destroyed, the page viewport is reset to its default value (see GpiCreatePS), and any clip region and path definition are lost. The save/restore presentation-space stack (see GpiSavePS) is purged.

Any palette selected into the presentation space remains selected.

Any dynamic segments left drawn on the device are not subsequently removed by GpiRemoveDynamics.

GpiAssociate - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_PS_IS_ASSOCIATED (0x20F5)
An attempt was made to destroy a presentation or associate a presentation space that is still associated with a device context.

PMERR_DC_IS_ASSOCIATED (0x2017)
An attempt was made to associate a presentation space with a device context that was already associated or to destroy a device context that was associated.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_HDC (0x207C)
An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

PMERR_REALIZE_NOT_SUPPORTED (0x20F7)
An attempt was made to create a realizable logical color table on a device driver that does not support this function.

PMERR_PATH_INCOMPLETE (0x20EC)
An attempt was made to open or close a segment either directly or during segment drawing, or to issue GpiAssociate while there is an open path bracket.

PMERR_AREA_INCOMPLETE (0x2005)
One of the following has occurred:
o A segment has been opened, closed, or drawn.
o GpiAssociate was issued while an area bracket was open.
o A drawn segment has opened an area bracket and ended without closing it.

GpiAssociate - Related Functions

Related Functions

- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSavePS
- GpiSetMarkerSet
- GpiSetPatternSet
- GpiSetPS

GpiAssociate - Example Code

This example releases the current device context and associates a new device context with the presentation space.

```
#define INCL_GPICONTROL         /* GPI control Functions        */
#include <os2.h>

HPS hps;                /* presentation space handle            */
HDC hdcPrinter;         /* device context handle                */

/* release the current device context */
GpiAssociate(hps, NULLHANDLE);
/* associate a printer device context */
```

```
GpiAssociate(hps, hdcPrinter);
```

GpiAssociate - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiBeginArea

GpiBeginArea - Syntax

This function begins the construction of an area.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
ULONG    flOptions;  /*  Area options. */
BOOL     rc;         /*  Success indicator. */

rc = GpiBeginArea(hps, flOptions);
```

GpiBeginArea Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiBeginArea Parameter - flOptions

flOptions (ULONG) - input
Area options.

This contains fields of option bits. One value should be selected for each option (unless the default is suitable). These values can be ORed together to determine whether to draw and include boundary lines, as well as the area interior.

Draw boundary lines:

BA_NOBOUNDARY	Does not draw boundary lines.
BA_BOUNDARY	Draws boundary lines (the default).

Construct interior mode:

BA_ALTERNATE	Constructs interior in *alternate* mode (the default).
BA_WINDING	Constructs interior in *winding* mode.

Include or exclude boundaries:

BA_INCL	Includes all boundaries in area (the default).
BA_EXCL	Excludes bottom and right boundaries.

GpiBeginArea Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE	Successful completion
FALSE	Error occurred.

GpiBeginArea - Parameters

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Area options.

This contains fields of option bits. One value should be selected for each option (unless the default is suitable). These values can be ORed together to determine whether to draw and include boundary lines, as well as the area interior.

Draw boundary lines:

BA_NOBOUNDARY	Does not draw boundary lines.
BA_BOUNDARY	Draws boundary lines (the default).

Construct interior mode:

BA_ALTERNATE	Constructs interior in *alternate* mode (the default).
BA_WINDING	Constructs interior in *winding* mode.

Include or exclude boundaries:

BA_INCL	Includes all boundaries in area (the default).
BA_EXCL	Excludes bottom and right boundaries.

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiBeginArea - Remarks

The construction is terminated by the GpiEndArea function.

You can use the following list of functions to define an area. They are used between the GpiBeginArea and GpiEndArea functions.

- GpiBeginElement
- GpiBox (with the *lControl* parameter set to DRO_OUTLINE)
- GpiCallSegmentMatrix
- GpiComment
- GpiElement (containing a valid call)
- GpiEndElement
- GpiFullArc (with the *lControl* parameter set to DRO_OUTLINE)
- GpiLabel
- GpiLine
- GpiMove
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolyLine
- GpiPolySpline
- GpiPop (that pops a valid call)
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs (setting valid line attributes only, or foreground color/mix (only) for other primitive types)
- GpiSetColor
- GpiSetCurrentPosition
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix
- GpiSetModelTransformMatrix

GpiBox and GpiFullArc are valid only in an area bracket (that is, between the GpiBeginArea and GpiEndArea functions with the *lControl* parameter set to DRO_OUTLINE. Other values of this parameter on these functions cause an implicit area bracket around the function.

Shading of the area is performed using the current pattern, as set by the GpiSetPattern function. The color and color-mixing modes that are current at the time GpiBeginArea is issued define the attributes to be applied to the pattern. The pattern reference point is also subjected to all of the transformations (including the model transformation) in force at the time of GpiBeginArea.

The area boundary consists of one or more *closed* figures, each constructed by:

- GpiBox
- GpiFullArc
- GpiPointArc
- GpiLine
- GpiPartialArc
- GpiPolyFilletSharp
- GpiPolyLine
- GpiPolySpline
- GpiPolyFillet

The GpiSetColor and GpiSetMix functions can be used to control how the area boundary is to be colored. The GpiSetLineEnd, GpiSetLineJoin, GpiSetLineType, and GpiSetLineWidth functions can be used to control line attributes as required. GpiSetAttrs can be used as an alternative way of setting these attributes. GpiSetArcParams can be used to control the shape of arcs produced by GpiFullArc, GpiPointArc, and GpiPartialArc.

The start of a new figure is indicated by:

- GpiCallSegmentMatrix
- GpiFullArc
- GpiMove
- GpiPop (or end of called segment), which pops current position or a model transform
- GpiSetCurrentPosition
- GpiSetModelTransformMatrix

Note: GpiCloseFigure must not be issued within an area.

A GpiBox or GpiFullArc function called within an area definition generates a complete closed figure. These functions must not be used within another figure definition.

The starting point of each closed figure is the current position when this function is made, or the point specified by the function starting the figure. Figure construction continues until either a new figure is started, or GpiEndArea is called.

Each figure should be closed, that is, the start and end points should be identical. If these points are not identical, they are joined by a straight line to arbitrarily close the figure.

The area interior is constructed either in *alternate* mode or in *winding* mode. In alternate mode, whether any point is within the interior is determined by drawing an imaginary line from that point to infinity; if there is an odd number of boundary crossings, the point is inside the area, if there is an even number of crossings, it is not.

In winding mode, the direction of the boundary lines is taken into account. Using the same imaginary line, the number of crossings is counted, as in alternate mode, but boundary lines going in one direction score plus one, and boundary lines going in the other direction score minus one. The point is in the interior if the final score is not zero.

In either mode, all of the boundaries of the area are considered to be part of the interior.

If the *flOptions* parameter of this function is BA_NOBOUNDARY, the boundary lines are *not* drawn, but the shading ends at the boundaries. If the *flOptions* parameter specifies BA_BOUNDARY the boundary lines and any lines added to close the figures are drawn. The lines are drawn using the current line attributes (which can be changed during construction) and shading occurs within the boundaries.

The current position is not changed by this function, but it can be changed by the moves, arcs, fillets, and lines between this function and the GpiEndArea function, including any used to close figures.

Area definitions cannot be nested. This function and the GpiEndArea function for one area must be within the same segment.

You can have no more than 1 450 straight-line vertices that describe the area.

During correlation in nonretained mode, a hit on any function within an area returns GPI_HITS in the GpiEndArea function. GPI_HITS is not returned on any of the primitives that occur within the area definition.

GpiBeginArea - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_AREA_CONTROL (0x2041)
: An invalid options parameter was specified with GpiBeginArea.

PMERR_INV_IN_PATH (0x208B)
: An attempt was made to issue a function invalid inside a path bracket.

PMERR_ALREADY_IN_AREA (0x2001)
: An attempt was made to begin a new area while an existing area bracket was already open.

GpiBeginArea - Related Functions

Related Functions

- GpiBeginPath
- GpiEndArea
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet

GpiBeginArea - Graphic Elements and Orders

Element Type: **OCODE_GBAR**

Order: Begin Area

GpiBeginArea - Example Code

This example uses the GpiBeginArea function to draw an area. The area, an isosceles triangle, is drawn with boundary lines and filled using the alternate filling mode.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>


HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiMove(hps, &ptlStart);    /* move to starting point (0, 0)     */
GpiBeginArea(hps,           /* start the area bracket            */
    BA_BOUNDARY |           /* draw boundary lines               */
    BA_ALTERNATE);          /* fill interior with alternate mode */
GpiPolyLine(hps, 3L, ptlTriangle);       /* draw the triangle    */
GpiEndArea(hps);                         /* end the area bracket */
```

GpiBeginArea - Topics

Select an item:
Syntax
Parameters
Returns
Errors

Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiBeginElement

GpiBeginElement - Syntax

This function defines the start of an element within a segment.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;      /*  Presentation-space handle. */
LONG    lType;    /*  Type to be associated with the element. */
PSZ     pszDesc;  /*  Description. */
BOOL    rc;       /*  Success indicator. */

rc = GpiBeginElement(hps, lType, pszDesc);
```

GpiBeginElement Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiBeginElement Parameter - lType

lType (LONG) - input
Type to be associated with the element.

Application-defined elements should have type values in the range 0x81xxxxxx through 0xFFxxxxxx to avoid conflict with system-generated elements.

GpiBeginElement Parameter - pszDesc

pszDesc (PSZ) - input
Description.

Variable-length character string, recorded with the type.

GpiBeginElement Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiBeginElement - Parameters

hps (HPS) - input
Presentation-space handle.

lType (LONG) - input
Type to be associated with the element.

Application-defined elements should have type values in the range 0x81xxxxxx through 0xFFxxxxxx to avoid conflict with system-generated elements.

pszDesc (PSZ) - input
Description.

Variable-length character string, recorded with the type.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiBeginElement - Remarks

This function starts an element, stored in the current segment, in **retain** or **draw-and-retain** mode (see GpiSetDrawingMode). The element is drawn in **draw** or **draw-and-retain** mode.

The drawing functions that form the contents of the element are passed on subsequent GPI functions (only those functions that can generate orders are logically part of the element). The element extends up to the next GpiEndElement function (or GpiCloseSegment, which causes an implicit GpiEndElement to be generated).

Grouping drawing functions together into an element is useful if the set of functions is to be changed or replaced together at a later time. Drawing functions that are not explicitly grouped together in an element bracket (GpiBeginElement - GpiEndElement pair) generate a single element for each GPI function.

The GpiElement function, that itself generates a complete element, is not allowed within an element bracket. The GpiLabel function is also not allowed within an element bracket. Elements must not be nested within one segment.

GpiBeginElement - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_ALREADY_IN_ELEMENT (0x2002)
An attempt was made to begin a new element while an existing element bracket was already open.

PMERR_DESC_STRING_TRUNCATED (0x2018)
An attempt was made to supply a description string with GpiBeginElement that was greater then the permitted maximum length (251 characters). The string was truncated.

GpiBeginElement - Related Functions

Related Functions

- GpiCloseSegment
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

GpiBeginElement - Graphic Elements and Orders

The element type is defined by the *lType* parameter.

Order: Begin Element

GpiBeginElement - Example Code

This example uses the GpiBeginElement function to create an element in a segment. The element type is 1 and the element description is "Triangle". The application can use these later to identify the element.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#include <os2.h>

HPS  hps;
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiBeginElement(hps,               /* start element bracket      */
    1L,                            /* element type is 1          */
    "Triangle");                   /* element description        */
GpiMove(hps, &ptlStart);           /* move to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw triangle              */
GpiEndElement(hps);                /* end element bracket        */
```

GpiBeginElement - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiBeginPath

GpiBeginPath - Syntax

This function specifies the start of a path.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lPath;  /*  Path identifier. */
BOOL    rc;     /*  Success indicator. */

rc = GpiBeginPath(hps, lPath);
```

GpiBeginPath Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiBeginPath Parameter - lPath

lPath (LONG) - input
Path identifier.

This must be 1.

GpiBeginPath Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiBeginPath - Parameters

hps (HPS) - input
Presentation-space handle.

lPath (LONG) - input
Path identifier.

This must be 1.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiBeginPath - Remarks

Paths can be used for these purposes:

- To generate lines and curves that have a geometric width (that is, a width that is subject to transformations); see GpiModifyPath and GpiStrokePath.

- To generate lines and curves that have cosmetic width; see GpiOutlinePath. In particular, if the lines and curves are defined by characters drawn with an outline font, hollow characters are produced. Hollow characters can also be drawn outside paths, using the FATTR_SEL_OUTLINE FATTRS option with the GpiCreateLogFont function.
- To generate nonrectangular shapes to be used for clipping; see GpiSetClipPath.
- To generate shapes to be filled; see GpiFillPath.

 Note: Areas can also be used for filling; see GpiBeginArea.
- To generate shapes to be converted to regions on which the region-combination function, GpiCombineRegion, can be used; see GpiPathToRegion.

There are two stages in the process of describing a path:

1. Path specification
2. Path definition.

Path Specification

A path is specified by a number of figures, within a GpiBeginPath - GpiEndPath pair. Each figure is specified by line functions, or curve functions, or both, and is separated from other figures by one of these functions:

- GpiCallSegmentMatrix
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiFullArc
- GpiMarker
- GpiMove
- GpiPolyMarker
- GpiPop (which restores the current position)
- GpiSetCurrentPosition
- GpiSetModelTransformMatrix

A figure that is terminated by one of the functions in this list is said to be an *open* figure. A figure can also be terminated by a GpiCloseFigure function. This is said to be a *closed* figure.

A GpiBox or GpiFullArc function within a path specifies a complete closed figure. These functions must not be used within another figure specification.

GpiBeginPath initializes the path to be empty.

Path specification functions are terminated by GpiEndPath. If there are no primitives between the GpiBeginPath and GpiEndPath functions, a null path is specified. The GpiEndPath that terminates this path specification must occur within the same segment as the GpiBeginPath function.

Path specification functions can occur within a segment bracket.

Path Definition

The process of path definition causes a description of the path to be built in the currently associated device context. This description is used during any subsequent operation on the path. If the definition occurred by the drawing of a retained segment containing specification functions, these may subsequently be edited, with no effect on the path definition, until the segment is drawn again.

If the drawing mode (see GpiSetDrawingMode) is set to **draw** or **draw-and-retain**, the path is defined as it is specified. If drawing mode is **retain**, path definition does not occur until the segment containing the path specification is drawn.

When a path has been defined, the definition cannot be reopened. An attempt to redefine the path results in the definition being replaced.

As the path definition is kept in the device context, association of the presentation space with a new device context means that the definition is lost.

When it has been defined, a path can be used only in a single GpiFillPath, GpiStrokePath, GpiOutlinePath, GpiPathToRegion, or GpiSetClipPath function. Alternatively, a path can be modified once only with a GpiModifyPath function, and then used in a single GpiFillPath, GpiPathToRegion, or GpiSetClipPath function. If a path is required to be reused in a normal (not a micro) presentation space, it can be created in a retained segment (for example, using **draw-and-retain** mode [see GpiSetDrawingMode]). This segment must be drawn whenever the definition has to be recreated. This may be done even if the application is otherwise nonretained. Otherwise, the application must reissue all the individual functions to reconstruct the path whenever the definition has to be recreated.

A path definition is bound in device coordinates at the time the path is defined. If any transforms (other than the final windowing transform) are subsequently changed, they have no effect on the path itself. However, they affect the thickness if the path is to be stroked using

GpiModifyPath, and they affect the pattern reference point if the path is to be filled with GpiFillPath. The transforms affect both the thickness and the pattern reference point if GpiStrokePath is used.

Other Remarks

Line type and line width have no effect on a path. Geometric line width takes effect if the path is stroked with GpiModifyPath or GpiStrokePath.

These functions can be used inside the path bracket (that is, between the GpiBeginPath function and the following GpiEndPath function) to define the path:

- GpiBeginElement (containing valid calls only)
- GpiBox (must specify DRO_OUTLINE option)
- GpiCallSegmentMatrix
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiCloseFigure
- GpiComment
- GpiElement (containing a valid call)
- GpiEndElement
- GpiFullArc (must specify DRO_OUTLINE option)
- GpiLabel
- GpiLine
- GpiMarker
- GpiMove
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolyMarker
- GpiPolyLine
- GpiPolySpline
- GpiPop (if only a valid call is popped)
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetCurrentPosition
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMarker
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetMix
- GpiSetModelTransformMatrix

The GpiCharString... functions, GpiQueryCharStringPos, GpiQueryCharStringPosAt, and GpiQueryTextBox are allowed only if the current font is an outline font.

You can have no more than 1 450 straight line vertices that describe the path. Curves are decomposed into straight lines internally, and the number of resulting vertices are also subject to this limit. The same applies to outline font character strings. If solid-filled outline characters are to be drawn, it is better to do this outside a path definition. GpiModifyPath and GpiStrokePath increase the number of lines in the path, and will cause a path initially containing more than 297 straight lines to exceed the limit of 1 450.

It is not valid for this function to occur within an area definition.

GpiBeginPath - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID (0x20AE)
An invalid path identifier parameter was specified.

PMERR_ALREADY_IN_PATH (0x2003)
An attempt was made to begin a new path while an existing path bracket was already open.

PMERR_INV_IN_AREA (0x2085)
An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

GpiBeginPath - Related Functions

Related Functions

- GpiBeginArea
- GpiCloseFigure
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetClipPath
- GpiSetColor
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMix
- GpiStrokePath

GpiBeginPath - Graphic Elements and Orders

Element Type: **OCODE_GBPTH**

Order: Begin Path

GpiBeginPath - Example Code

This example uses the GpiBeginPath function to create a path. The path, an isosceles triangle, is given path identifier 1. After the path bracket is ended using GpiEndPath, a subsequent call to the GpiFillPath function draws and fills the path.

```
#define INCL_GPIPATHS           /* GPI Path functions           */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiBeginPath(hps, 1L);                /* start the path bracket */
GpiMove(hps, &ptlStart);              /* move to starting point */
GpiPolyLine(hps, 2L, ptlTriangle);    /* draw two sides         */
GpiCloseFigure(hps);                  /* close the triangle     */
GpiEndPath(hps);                      /* end the path bracket   */
GpiFillPath(hps, 1L, FPATH_ALTERNATE); /* draw and fill the path */
```

GpiBeginPath - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiBitBlt

GpiBitBlt - Syntax

This function copies a rectangle of bit-map image data.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hpsTarget;   /*  Target presentation-space handle. */
HPS        hpsSource;   /*  Source presentation-space handle. */
LONG       lCount;      /*  Point count. */
PPOINTL    aptlPoints;  /*  Point array. */
LONG       lRop;        /*  Mixing function required. */
ULONG      flOptions;   /*  Options. */
LONG       lHits;       /*  Correlation and error indicators. */

lHits = GpiBitBlt(hpsTarget, hpsSource, lCount,
          aptlPoints, lRop, flOptions);
```

GpiBitBlt Parameter - hpsTarget

hpsTarget (HPS) - input
Target presentation-space handle.

GpiBitBlt Parameter - hpsSource

hpsSource (HPS) - input
Source presentation-space handle.

GpiBitBlt Parameter - lCount

lCount (LONG) - input
Point count.

Number of points specified in *aptlPoints* .

It must be 3 or 4. If this is 3, a source rectangle of the same size as the target rectangle is used. If it is 4, stretching or compression is performed as necessary. If compression is performed, the *flOptions* parameter determines how eliminated rows or columns are handled.

GpiBitBlt Parameter - aptlPoints

aptlPoints (PPOINTL) - input
Point array.

Array of *lCount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**, where:

Tx1,Ty1	Specify the lower-left corner of the target rectangle in target device coordinates.
Tx2,Ty2	Specify the upper-right corner of the target rectangle in target device coordinates.
Sx1,Sy1	Specify the lower-left corner of the source rectangle in source device coordinates.
Sx2,Sy2	Specify the upper-right corner of the source rectangle in source device coordinates (not required if neither stretching nor compression is to be performed).

GpiBitBlt Parameter - lRop

lRop (LONG) - input

Mixing function required.

The value of *lRop* required to achieve any given result can be determined from the following table. The final value of each bit in every pel depends on the values of the corresponding bits in the pattern (P), source (S), and the original target value (T initial). Each row of the table shows one of the 8 possible combinations of these values. For each combination, mark the desired final target value in the last column. The 8 bits in this column then show the value of the least significant byte of *lRop* required to achieve this mixing function. For example, if the required mixing function is to copy the source to the target, then the T (final) column will be the same as the S column, and so *lRop* will have the binary value 11001100, or the hexadecimal value 00CC.

```
P       S       T (initial)      T (final)

0       0       0                Bit 0 (least significant)

0       0       1                Bit 1

0       1       0                Bit 2

0       1       1                Bit 3

1       0       0                Bit 4

1       0       1                Bit 5

1       1       0                Bit 6

1       1       1                Bit 7 (most significant)
```

Mnemonic names are available for commonly used mixes:

```
ROP_SRCCOPY       /* SRC                     */
ROP_SRCPAINT      /* SRC OR DST              */
ROP_SRCAND        /* SRC AND DST             */
ROP_SRCINVERT     /* SRC XOR DST             */
ROP_SRCERASE      /* SRC AND NOT(DST)        */
ROP_NOTSRCCOPY    /* NOT(SRC)                */
ROP_NOTSRCERASE   /* NOT(SRC) AND NOT(DST)   */
ROP_MERGECOPY     /* SRC AND PAT             */
ROP_MERGEPAINT    /* NOT(SRC) OR DST         */
ROP_PATCOPY       /* PAT                     */
ROP_PATPAINT      /* NOT(SRC) OR PAT OR DST  */
ROP_PATINVERT     /* DST XOR PAT             */
ROP_DSTINVERT     /* NOT(DST)                */
ROP_ZERO          /* 0                       */
ROP_ONE           /* 1                       */
```

GpiBitBlt Parameter - flOptions

flOptions (ULONG) - input
Options.

The options define how eliminated lines or columns are treated if a compression is performed.

Bits 15 through 31 of *flOptions* may be used for privately supported modes for particular devices.

BBO_OR
: The default. If compression is necessary, logical-OR the eliminated rows or columns. This is useful for white on black.

BBO_AND
: If compression is necessary, logical-AND the eliminated rows or columns. This is useful for black on white.

BBO_IGNORE
: If compression is necessary, ignore the eliminated rows or columns. This is useful for color.

BBO_PAL_COLORS
: The color table passed in with the BitmapInfoTable is defined as indices into the currently realized palette, rather than actual colors.

GpiBitBlt Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful completion

GPI_HITS
: Correlate hits

GPI_ERROR
: Error occurred.

GpiBitBlt - Parameters

hpsTarget (HPS) - input
Target presentation-space handle.

hpsSource (HPS) - input
Source presentation-space handle.

lCount (LONG) - input
Point count.

Number of points specified in *aptlPoints* .

It must be 3 or 4. If this is 3, a source rectangle of the same size as the target rectangle is used. If it is 4, stretching or compression is performed as necessary. If compression is performed, the *flOptions* parameter determines how eliminated rows or columns are handled.

aptlPoints (PPOINTL) - input
Point array.

Array of *lCount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**, where:

Tx1,Ty1	Specify the lower-left corner of the target rectangle in target device coordinates.
Tx2,Ty2	Specify the upper-right corner of the target rectangle in target device coordinates.
Sx1,Sy1	Specify the lower-left corner of the source rectangle in source device coordinates.
Sx2,Sy2	Specify the upper-right corner of the source rectangle in source device coordinates (not required if neither stretching nor compression is to be performed).

lRop (LONG) - input
Mixing function required.

The value of *lRop* required to achieve any given result can be determined from the following table. The final value of each bit in every pel depends on the values of the corresponding bits in the pattern (P), source (S), and the original target value (T initial). Each row of the table shows one of the 8 possible combinations of these values. For each combination, mark the desired final target value in the last column. The 8 bits in this column then show the value of the least significant byte of *lRop* required to achieve this mixing function. For example, if the required mixing function is to copy the source to the target, then the T (final) column will be the same as the S column, and so *lRop* will have the binary value 11001100, or the hexadecimal value 00CC.

```
P       S       T (initial)      T (final)
```

```
0       0       0               Bit 0 (least significant)

0       0       1               Bit 1

0       1       0               Bit 2

0       1       1               Bit 3

1       0       0               Bit 4

1       0       1               Bit 5

1       1       0               Bit 6

1       1       1               Bit 7 (most significant)
```

Mnemonic names are available for commonly used mixes:

```
ROP_SRCCOPY       /* SRC                     */
ROP_SRCPAINT      /* SRC OR DST              */
ROP_SRCAND        /* SRC AND DST             */
ROP_SRCINVERT     /* SRC XOR DST             */
ROP_SRCERASE      /* SRC AND NOT(DST)        */
ROP_NOTSRCCOPY    /* NOT(SRC)                */
ROP_NOTSRCERASE   /* NOT(SRC) AND NOT(DST)   */
ROP_MERGECOPY     /* SRC AND PAT             */
ROP_MERGEPAINT    /* NOT(SRC) OR DST         */
ROP_PATCOPY       /* PAT                     */
ROP_PATPAINT      /* NOT(SRC) OR PAT OR DST  */
ROP_PATINVERT     /* DST XOR PAT             */
ROP_DSTINVERT     /* NOT(DST)                */
ROP_ZERO          /* 0                       */
ROP_ONE           /* 1                       */
```

flOptions (ULONG) - input
Options.

The options define how eliminated lines or columns are treated if a compression is performed.

Bits 15 through 31 of *flOptions* may be used for privately supported modes for particular devices.

BBO_OR
: The default. If compression is necessary, logical-OR the eliminated rows or columns. This is useful for white on black.

BBO_AND
: If compression is necessary, logical-AND the eliminated rows or columns. This is useful for black on white.

BBO_IGNORE
: If compression is necessary, ignore the eliminated rows or columns. This is useful for color.

BBO_PAL_COLORS
: The color table passed in with the BitmapInfoTable is defined as indices into the currently realized palette, rather than actual colors.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful completion

GPI_HITS
: Correlate hits

GPI_ERROR
: Error occurred.

GpiBitBlt - Remarks

A rectangle of bit-map image data is copied from a bit map selected into a device context associated with the source presentation space, to a bit map selected into a device context associated with the target presentation space. Alternatively, either presentation space may be associated with a device context that specifies a suitable raster device, for example, the screen.

Note: In either case, both source and target device contexts must apply to the same physical device. It is an error if this device does not support raster operations.

Unless the device is a *banded* printer, both source and target may refer to the same presentation space. If so, the copy is nondestructive when source and target rectangles overlap.

A rectangle can be specified in device coordinates, for both source and target. These rectangles are noninclusive; that is, they include the left and lower boundaries in device space, but not the right and upper boundaries. Thus, if the lower-left maps to the same device pel as the upper-right, that rectangle is considered to be empty.

If the upper-right source point is specified, and the source and target rectangles are of different sizes, stretching, or compressing, or both, of the data occurs. *flOptions* specifies how eliminated rows or columns of bits are to be treated if compression occurs. Note that the pattern data is never stretched or compressed.

The following current attributes of the target presentation space are used (other than for converting between monochrome and color, as described below):

- Area color
- Area background color
- Pattern set
- Pattern symbol.

The color values are used in conversion between monochrome and color data. This is the only format conversion performed by this function. The conversions are:

- Output of a monochrome pattern to a color device.

 In this instance, the pattern is converted first to a color pattern using the current area colors:

 - source 1s -> area foreground color
 - source 0s -> area background color.

- Copying from a monochrome bit map to a color bit map (or device).

 The source bits are converted as follows:

 - source 1s -> image foreground color
 - source 0s -> image background color.

- Copying from a color bit map to a monochrome bit map (or device).

 - The image will be grey scaled under the current operation of the device drivers, and the colors will be shades of grey corresponding to a mapping scheme that the device driver uses. This behavior may vary from one printer to another, depending on the printer's capabilities.

 The only variance from this grey-scaling is that pure white (for example, backgrounds) prints as black and pure black (for example, text) prints as white.

Note: In all of the above instances (except where the source image background color is used) it is the attributes of the *target* presentation space that are used.

If the mix (*lRop*) does not call for a pattern, the pattern set and pattern symbol are not used. If it does not require a source (this is not valid when *flOptions* is in the range 1 through 3), *hpsSource* is not required and must be null. **Sx1,Sy1** is also ignored in this instance.

Neither the source nor the pattern is required when a bit map, or part of a bit map, is to be cleared to a particular color.

If the mix does require both source and pattern, a three-way operation is performed.

If a pattern is required, dithering may be performed for solid patterns in a color that is not available on the device; see GpiSetPattern.

Support for the BM_SRCTRANSPARENT and BM_DESTTRANSPARENT background mix options is provided by the CAPS_BACKGROUND_MIX_SUPPORT flags in DevQueryCaps.

Requesting the BM_SRCTRANSPARENT background mix option results in pels from the source bit map matching the presentaton space background color, to not be copied to the output bit map, effectively leaving those pels in the output unchanged. This provides for a transparent overlay function.

Requesting the BM_DESTTRANSPARENT background mix option results in a transfer such that pels from the source bit map will only be copied to destination pels that match the presentation space background color. This provides for a transparent underlay function.

If any of the source data is not available (when, for example, the source presentation space is connected to a screen window, and the source rectangle is not totally visible), the contents of the unavailable parts are undefined. This can be checked with GpiRectVisible before calling this function.

This function is independent of drawing mode (see GpiSetDrawingMode); the effect always occurs immediately, and it is not retained even if the drawing mode is **draw-and-retain** or **retain**. Its effect, however, is recorded in a metafile, but note that this is successful only if the metafile is replayed on a similar device, with **draw** drawing mode.

The current position in both source and target presentation spaces is unchanged by this function.

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiBitBlt - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_BITBLT_MIX (0x2046)
An invalid *lRop* was specified with a GpiBitBlt or GpiWCBitBlt function.

PMERR_INV_BITBLT_STYLE (0x2047)
An invalid options parameter was specified with a GpiBitBlt or GpiWCBitBlt function.

PMERR_BITMAP_NOT_FOUND (0x200A)
A attempt was made to perform a bit-map operation on a bit map that did not exist.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
An invalid rectangle parameter was specified.

PMERR_NO_BITMAP_SELECTED (0x20E4)
An attempt has been made to operate on a memory device context that has no bit map selected.

PMERR_INCORRECT_DC_TYPE (0x203C)
An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

PMERR_INCOMPATIBLE_BITMAP (0x203A)
An attempt was made to select a bit map or perform a BitBlt operation on a device context that was incompatible with the format of the bit map.

GpiBitBlt - Related Functions

Related Functions

- GpiCreateBitmap
- GpiDeleteBitmap

- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt

GpiBitBlt - Example Code

This example uses GpiBitBlt to copy a bit map from one presentation space to another. Two presentation spaces are created: one associated with a memory context, and the other associated with a screen context. The function copies the memory context bit map that is 100 pels wide and 100 pels high into a 50-by-50-pel rectangle at the location (300,400) on the screen, thereby causing the bit map to be visible in the window. Since the raster operation is ROP_SRCCOPY, GpiBitBlt replaces the image previously in the target rectangle. The function compresses the bit map to fit the new rectangle by discarding extra rows and columns as specified by the BBO_IGNORE option.

```
#define INCL_GPIBITMAPS         /* Bit map functions            */
#define INCL_DEV                /* Device Function definitions  */
#define INCL_GPICONTROL         /* GPI control Functions        */
#define INCL_WINWINDOWMGR       /* Window Manager Functions     */
#include <os2.h>

HAB     hab;            /* anchor-block handle                  */
HPS    hpsMemory;       /* presentation-space handle            */
HPS    hpsScreen;       /* presentation-space handle            */
HDC    hdcScreen;       /* Device-context handle                */
HDC    hdcMemory;       /* Device-context handle                */
SIZEL  sizl={0, 0};     /* use same page size as device         */
/* context data structure */
DEVOPENSTRUC dop = {0L, "DISPLAY", NULL, 0L, 0L, 0L, 0L, 0L};
POINTL aptl[4] = {
    300, 400,           /* lower-left corner of target          */
    350, 450,           /* upper-right corner of target         */
    0, 0,               /* lower-left corner of source          */
    100, 100 };         /* upper-right corner of source         */
HWND   hwnd;

/* create memory device context and presentation space, associating
   DC with the PS */
hdcMemory = DevOpenDC(hab, OD_MEMORY, "*", 5L, (PDEVOPENDATA)&dop,
                      NULLHANDLE);
hpsMemory = GpiCreatePS(hab, hdcMemory, &sizl, GPIA_ASSOC
                                             | PU_PELS);

/* create window device context and presentation space, associating
   DC with the PS */
hdcScreen = WinOpenWindowDC(hwnd); /* Open window device context */
hpsScreen = GpiCreatePS(hab, hdcScreen, &sizl, PU_PELS | GPIF_LONG
                                             | GPIA_ASSOC);


/*
  .
  . get bit map, associate bit map with memory device context,
  .     draw into bit map
  .
  */

/* display the bit map on the screen by copying it from the memory
   device context into the screen device context */
GpiBitBlt(hpsScreen, hpsMemory, 4L, aptl, ROP_SRCCOPY, BBO_IGNORE);
```

GpiBitBlt - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiBox

GpiBox - Syntax

This function draws a rectangular box with the current position and a specified position at diagonally opposite corners.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
LONG       lControl;   /*  Outline and fill control. */
PPOINTL    pptlPoint;  /*  Corner point. */
LONG       lHRound;    /*  Corner-rounding control. */
LONG       lVRound;    /*  Corner-rounding control. */
LONG       lHits;      /*  Correlation and error indicators. */

lHits = GpiBox(hps, lControl, pptlPoint, lHRound,
          lVRound);
```

GpiBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiBox Parameter - lControl

lControl (LONG) - input

Outline and fill control.

Specifies if the interior of the box is to be filled, and if the outline is to be drawn. This parameter can have one of the following values:

DRO_FILL
: Fill interior

DRO_OUTLINE
: Draw outline

DRO_OUTLINEFILL
: Draw outline and fill interior.

GpiBox Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Corner point.

The coordinates of the corner that is diagonally opposite to the current position.

GpiBox Parameter - lHRound

lHRound (LONG) - input
Corner-rounding control.

Horizontal length of the *full* axis of the ellipse that is used for rounding at each corner.

GpiBox Parameter - lVRound

lVRound (LONG) - input
Corner-rounding control.

Vertical length of the *full* axis of the ellipse that is used for rounding at each corner.

GpiBox Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiBox - Parameters

hps (HPS) - input
Presentation-space handle.

lControl (LONG) - input
Outline and fill control.

Specifies if the interior of the box is to be filled, and if the outline is to be drawn. This parameter can have one of the following values:

DRO_FILL
Fill interior
DRO_OUTLINE
Draw outline
DRO_OUTLINEFILL
Draw outline and fill interior.

pptlPoint (PPOINTL) - input
Corner point.

The coordinates of the corner that is diagonally opposite to the current position.

lHRound (LONG) - input
Corner-rounding control.

Horizontal length of the *full* axis of the ellipse that is used for rounding at each corner.

lVRound (LONG) - input
Corner-rounding control.

Vertical length of the *full* axis of the ellipse that is used for rounding at each corner.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiBox - Remarks

The sides of the box are parallel to the world coordinate x- and y-axes.

The four corners of the box can be rounded with a quarter ellipse. The size of this ellipse is specified by *lHRound* and *lVRound*. If *lHRound* equals *lVRound*, the corners of the box are rounded with a quarter circle.

If either *lHRound* or *lVRound* is zero, no rounding occurs.

If the current position is (x0,y0) and *pptlPoint* is set to (x1,y1), the box is drawn from (x0,y0) to (x1,y0) to (x1,y1) to (x0,y1) to (x0,y0). The direction of drawing is significant in area winding mode; see GpiBeginArea.

The current position is unchanged by this function.

Either the outline of the box, or its interior, or both, can be drawn.

If this function occurs within an area or path definition, it generates a complete closed figure (DRO_OUTLINE must be specified). It must not occur within any other figure definition.

If correlation is in force, a hit always results if the pick aperture intersects the box boundary. However, if the pick aperture lies wholly within the box, a hit only occurs if the interior is being drawn (DRO_FILL or DRO_OUTLINEFILL).

GpiBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_BOX_CONTROL (0x2049)
An invalid control parameter was specified with GpiBox.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_BOX_ROUNDING_PARM (0x204A)
An invalid corner rounding control parameter was specified with GpiBox.

GpiBox - Related Functions

Related Functions

- GpiBox
- GpiPop
- GpiQueryCurrentPosition
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix
- GpiSetCurrentPosition
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom

GpiBox - Graphic Elements and Orders

Element Type: **OCODE_GCBOX**

Order: Box at Given Position

GpiBox - Example Code

This example calls GpiBox to draw a series of rounded boxes, one inside another.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptl = { 100, 100 };
SHORT i;

for (i = 0; i < 5; i++)
    GpiBox(hps,             /* handle to a presentation space    */
        DRO_OUTLINE,        /* draw the box outline  */
        &ptl,               /* address of the corner */
        i * 10L,            /* horizontal corner radius */
        i * 10L);           /* vertical corner radius      */
```

GpiBox - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiCallSegmentMatrix

GpiCallSegmentMatrix - Syntax

This function calls a segment and applies an instance transform to it.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;           /*  Presentation-space handle. */
LONG         lSegment;      /*  Identifier of segment to be called. */
LONG         lCount;        /*  Number of elements. */
PMATRIXLF    pmatlfArray;   /*  Instance transform matrix. */
LONG         lOptions;      /*  Transformation options. */
LONG         lHits;         /*  Correlation and error indicators. */

lHits = GpiCallSegmentMatrix(hps, lSegment,
          lCount, pmatlfArray, lOptions);
```

GpiCallSegmentMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCallSegmentMatrix Parameter - lSegment

lSegment (LONG) - input
Identifier of segment to be called.

This must be greater than 0.

The segment must not be a chained segment.

GpiCallSegmentMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements of *pmatlfArray* to be examined, starting from the beginning of the structure. If *lCount* is less or equal than 9, the remaining elements default to the corresponding elements of the identity matrix. If *lCount* = 0, the identity matrix is used.

GpiCallSegmentMatrix Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - input
Instance transform matrix.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

GpiCallSegmentMatrix Parameter - lOptions

lOptions (LONG) - input
Transformation options.

Specify how the transform defined by the *pmatlfArray* parameter should be used to modify the existing current model transform for

the duration of the function. The existing transform is the concatenation, in the current function context, of the instance, segment, and model transforms, from the root segment downwards.

TRANSFORM_REPLACE
: The previous model transform is discarded and replaced by the specified transform.

TRANSFORM_ADD
: The specified transform is combined with the existing model transform. The existing transform precedes the new transform. This option is most useful for incremental updates to transforms.

TRANSFORM_PREEMPT
: The specified transform is combined with the existing model transform. The new transform precedes the existing transform.

GpiCallSegmentMatrix Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiCallSegmentMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lSegment (LONG) - input
Identifier of segment to be called.

This must be greater than 0.

The segment must not be a chained segment.

lCount (LONG) - input
Number of elements.

The number of elements of *pmatlfArray* to be examined, starting from the beginning of the structure. If *lCount* is less or equal than 9, the remaining elements default to the corresponding elements of the identity matrix. If *lCount* = 0, the identity matrix is used.

pmatlfArray (PMATRIXLF) - input
Instance transform matrix.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

lOptions (LONG) - input
Transformation options.

Specify how the transform defined by the *pmatlfArray* parameter should be used to modify the existing current model transform for the duration of the function. The existing transform is the concatenation, in the current function context, of the instance, segment, and model transforms, from the root segment downwards.

TRANSFORM_REPLACE
: The previous model transform is discarded and replaced by the specified transform.

TRANSFORM_ADD
: The specified transform is combined with the existing model transform. The existing transform precedes the new

transform. This option is most useful for incremental updates to transforms.

TRANSFORM_PREEMPT
: The specified transform is combined with the existing model transform. The new transform precedes the existing transform.

lHits (LONG) - returns
: Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiCallSegmentMatrix - Remarks

The instance transform specified is a model transform that is used to modify the current model transform, in a way that depends upon the value of the *lOptions* parameter, before calling the segment. This new transform applies only to the called segment. On return, it is reset to the model transform in operation before the function was called.

The transform is specified as a one-dimensional array of elements, being the first *lCount* elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

```
   Matrix                   Array


  a  b  0
  c  d  0            (a,b,0,c,d,0,e,f,1)
  e  f  1
```

A point with coordinates (x,y) is transformed to the point

```
(a*x + c*y + e, b*x + d*y + f)
```

The called segment must have a unity transform for the viewing transform (see GpiSetViewingTransformMatrix).

If scaling values greater than unity are given (which only applies if the presentation space coordinate format as set by the GpiCreatePS function is GPIF_LONG), it is possible for the combined effect of this and any other relevant transforms to exceed fixed-point implementation limits. This causes an error.

GpiCallSegmentMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
: An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_MATRIX_ELEMENT (0x209B)
An invalid transformation matrix element was specified.

PMERR_INV_TRANSFORM_TYPE (0x20D0)
An invalid options parameter was specified with a transform matrix function.

PMERR_CALLED_SEG_NOT_FOUND (0x200F)
An attempt was made to call a segment that did not exist.

PMERR_CALLED_SEG_IS_CHAINED (0x200D)
An attempt was made to call a segment that has a chained attribute set.

PMERR_CALLED_SEG_IS_CURRENT (0x200E)
An attempt was made to call a segment that is currently open.

PMERR_SEG_CALL_STACK_EMPTY (0x20FC)
A call stack empty condition was detected when attempting a pop function during GpiPop or segment drawing.

GpiCallSegmentMatrix - Related Functions

Related Functions

- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority
- GpiSetSegmentTransformMatrix

GpiCallSegmentMatrix - Graphic Elements and Orders

Element Type: **OCODE_GCALLS**

Order: Push and Set Model Transform

Order: Call Segment

Order: Pop

GpiCallSegmentMatrix - Example Code

This example calls the GpiCallSegmentMatrix function to draw a segment three times. Each time the segment is drawn, the instance transformation doubles in size. The result is three triangles with the last triangle twice the size of the second, and the second twice the size of the first.

```
#define INCL_GPITRANSFORMS      /* GPI Transform functions      */
#define INCL_GPISEGMENTS        /* Segment functions            */
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS    hps;
USHORT i;
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */
MATRIXLF matlfInstance = { MAKEFIXED(1, 0),  MAKEFIXED(0, 0), 0,
                           MAKEFIXED(0, 0),  MAKEFIXED(1, 0), 0,
                           0,                0,               1 };

GpiOpenSegment(hps, 1L);             /* opens segment               */
GpiMove(hps, &ptlStart);             /* moves to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle);   /* draws triangle              */
GpiCloseSegment(hps);                /* closes segment              */

for (i = 0; i < 3; i++)
    {
    /*
     * Draw the segment after adding the matrix to the model
     * transformation.
     */

    GpiCallSegmentMatrix(hps, 1L, 9, &matlfInstance, TRANSFORM_ADD);
    matlfInstance.fxM11 *= 2;
    matlfInstance.fxM22 *= 2;
    }
```

GpiCallSegmentMatrix - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiCharString

GpiCharString - Syntax

This function draws a character string starting at the current position.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;        /*  Presentation-space handle. */
LONG    lCount;     /*  Number of bytes in the string. */
PCH     pchString;  /*  Characters to be drawn. */
LONG    lHits;      /*  Correlation and error indicators. */

lHits = GpiCharString(hps, lCount, pchString);
```

GpiCharString Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCharString Parameter - lCount

lCount (LONG) - input
Number of bytes in the string.

Must be greater than 0 and less or equal to 512.

GpiCharString Parameter - pchString

pchString (PCH) - input
Characters to be drawn.

This parameter does not need to be null terminated.

GpiCharString Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiCharString - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of bytes in the string.

Must be greater than 0 and less or equal to 512.

pchString (PCH) - input
Characters to be drawn.

This parameter does not need to be null terminated.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiCharString - Remarks

Each character in the string is positioned so that its character reference point is at the current position. The current position is advanced after each character is drawn to give the position for the next character.

The characters in the character string are selected from the current character set. The font from which the characters are selected depends on the current character mode. For a description of which fonts are used for each of the possible modes, see GpiSetCharMode.

The degree to which approximation of the position and size of characters is allowed, and also the area used during correlation of the character string, is controlled by the character-mode attribute.

After the string has been drawn, the current position is set to the end of the character string. This is the point at which the next character would have been drawn, had it existed.

GpiCharString - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_FONT_AND_MODE_MISMATCH (0x202D)
An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.

GpiCharString - Related Functions

Related Functions

- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiPop
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryDefCharBox
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix

GpiCharString - Graphic Elements and Orders

Element Type: **OCODE_GCCHSTM**

Order: Character String Move at Current Position

GpiCharString - Example Code

This example uses the GpiCharString function to draw the string "Hello". The GpiMove function moves the current position to (100,100) so that the string starts there.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart;        /* beginning of string                  */

ptlStart.x = 100L;
ptlStart.y = 100L;

/* Start string at (100, 100). */

GpiMove(hps, &ptlStart);

/* Draw the 5-character string. */
```

```
GpiCharString(hps, 5L, "Hello");
```

GpiCharString - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiCharStringAt

GpiCharStringAt - Syntax

This function draws a character string starting at a specified position.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlPoint;  /*  Starting position. */
LONG       lCount;     /*  Number of bytes in the string. */
PCH        pchString;  /*  Characters to be drawn. */
LONG       lHits;      /*  Correlation and error indicators. */

lHits = GpiCharStringAt(hps, pptlPoint, lCount,
          pchString);
```

GpiCharStringAt Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCharStringAt Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Starting position.

Defines, in world coordinates, the position at which the first character in the string is to be placed.

GpiCharStringAt Parameter - lCount

lCount (LONG) - input
Number of bytes in the string.

Must be greater than 0 and less or equal to 512.

GpiCharStringAt Parameter - pchString

pchString (PCH) - input
Characters to be drawn.

This parameter does not need to be null terminated.

GpiCharStringAt Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiCharStringAt - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - input
Starting position.

Defines, in world coordinates, the position at which the first character in the string is to be placed.

lCount (LONG) - input
Number of bytes in the string.

Must be greater than 0 and less or equal to 512.

pchString (PCH) - input
Characters to be drawn.

This parameter does not need to be null terminated.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiCharStringAt - Remarks

The function GpiCharStringAt (hps, point, count, string) is equivalent to:

```
GpiMove (hps, point)
GpiCharString (hps, count, string)
```

Each character in the string is positioned so that its character reference point is at the current position. The current position is advanced after each character is drawn to give the position for the next character.

The font from which the characters in the character string are selected depends on the current character mode. For a description of which fonts are used for each of the possible modes, see GpiSetCharMode.

The degree to which approximation of the position and size is allowed, and also the area used during correlation of the character string, is controlled by the character-mode attribute.

After the string has been drawn, the current position is set to the end of the character string. This is the point at which the next character would have been drawn, had it existed.

GpiCharStringAt - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_FONT_AND_MODE_MISMATCH (0x202D)
An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.

GpiCharStringAt - Related Functions

Related Functions

- GpiCharString
- GpiCharStringPos
- GpiCharStringPosAt
- GpiPop
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryDefCharBox
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix

GpiCharStringAt - Graphic Elements and Orders

Element Type: **OCODE_GCHSTM**

Order: Character String Move at Given Position

GpiCharStringAt - Example Code

This example uses the GpiCharStringAt function to draw the string "Hello" starting at the position (100,100). It then uses the GpiMove and GpiCharString functions to draw the same string at exactly the same position.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart;

ptlStart.x = 100L;
ptlStart.y = 100L;

/* Draw the string "Hello" at (100, 100). */

GpiCharStringAt(hps, &ptlStart, 5, "Hello");

/* These two calls are identical to the one above. */

GpiMove(hps, &ptlStart);
GpiCharString(hps, 5L, "Hello");
```

GpiCharStringAt - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiCharStringPos

GpiCharStringPos - Syntax

This function draws a character string starting at the current position, with formatting options.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
PRECTL    prclRect;   /*  Rectangle structure. */
ULONG     flOptions;  /*  Formatting options. */
LONG      lCount;     /*  Number of bytes in the string. */
PCH       pchString;  /*  Characters to be drawn. */
PLONG     alAdx;      /*  Increment values. */
LONG      lHits;      /*  Correlation and error indicators. */

lHits = GpiCharStringPos(hps, prclRect, flOptions,
          lCount, pchString, alAdx);
```

GpiCharStringPos Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCharStringPos Parameter - prclRect

prclRect (PRECTL) - input
Rectangle structure.

Defines, in world coordinates, the two corners of the rectangle that defines the background of the characters. It is ignored unless CHS_OPAQUE or CHS_CLIP is specified.

GpiCharStringPos Parameter - flOptions

flOptions (ULONG) - input
Formatting options.

Option flags that can be used in combination:

CHS_OPAQUE
Background of characters is defined by the rectangle specified by *prclRect*. The rectangle is to be shaded (with background color and overpaint) before drawing.

CHS_VECTOR
Increments vector (*alAdx*) is supplied. If zero, *alAdx* is ignored.

CHS_LEAVEPOS
Leave the current position at the start of the string. If not set, the current position is moved to the position at which the next character would have been drawn, had there been one.

CHS_CLIP
Clip the string to the rectangle.

CHS_UNDERSCORE
Underscore the characters. See FATTR_SEL_UNDERSCORE field in the FATTRS datatype.

CHS_STRIKEOUT
Overstrike the characters. See FATTR_SEL_STRIKEOUT field in the FATTRS datatype.

Other bits are reserved and must be zero.

GpiCharStringPos Parameter - lCount

lCount (LONG) - input
Number of bytes in the string.

Must be greater than 0 and less or equal to 512.

GpiCharStringPos Parameter - pchString

pchString (PCH) - input
Characters to be drawn.

This parameter does not need to be null terminated.

GpiCharStringPos Parameter - alAdx

alAdx (PLONG) - input
Increment values.

Vector of increment values, in world coordinates. Any negative values are treated as if they were zero.

GpiCharStringPos Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiCharStringPos - Parameters

hps (HPS) - input
Presentation-space handle.

prclRect (PRECTL) - input
Rectangle structure.

Defines, in world coordinates, the two corners of the rectangle that defines the background of the characters. It is ignored unless CHS_OPAQUE or CHS_CLIP is specified.

flOptions (ULONG) - input
Formatting options.

Option flags that can be used in combination:

CHS_OPAQUE
: Background of characters is defined by the rectangle specified by *prclRect*. The rectangle is to be shaded (with background color and overpaint) before drawing.

CHS_VECTOR
: Increments vector (*alAdx*) is supplied. If zero, *alAdx* is ignored.

CHS_LEAVEPOS
: Leave the current position at the start of the string. If not set, the current position is moved to the position at which the next character would have been drawn, had there been one.

CHS_CLIP
: Clip the string to the rectangle.

CHS_UNDERSCORE

Underscore the characters. See FATTR_SEL_UNDERSCORE field in the FATTRS datatype.

CHS_STRIKEOUT
Overstrike the characters. See FATTR_SEL_STRIKEOUT field in the FATTRS datatype.

Other bits are reserved and must be zero.

lCount (LONG) - input
Number of bytes in the string.

Must be greater than 0 and less or equal to 512.

pchString (PCH) - input
Characters to be drawn.

This parameter does not need to be null terminated.

alAdx (PLONG) - input
Increment values.

Vector of increment values, in world coordinates. Any negative values are treated as if they were zero.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiCharStringPos - Remarks

A vector of increments can be specified, allowing control over the positioning of each character after the first. This vector consists of distances measured in world coordinates (along the baseline for left-to-right and right-to-left character directions, and along the shearline for top-to-bottom and bottom-to-top character directions). Increment i is the distance of the reference point of character $i+1$ from the reference point of character i. The last increment may be needed to update the current position.

These increments, when specified, set the widths of each character.

A further option allows a rectangle to be specified that can be used as the background of the string instead of the normal background. This rectangle is painted using the current character background color and an overpaint mix (unless this is in a dynamic segment, when leave-alone is used). Both corners of the rectangle are specified, so that the rectangle is positioned independently of the current position. Points on the borders of the rectangle are considered to be included within the rectangle.

Clipping of the string to the rectangle is also allowed. This is independent of whether the rectangle is actually drawn.

The current position can be updated to the point at which the next character would have been drawn, had there been one, or it can be left at the start of the string.

GpiCharStringPos - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_POS_OPTIONS (0x204E)
An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_RECT (0x20BD)
An invalid rectangle parameter was specified.

PMERR_FONT_AND_MODE_MISMATCH (0x202D)
An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.

GpiCharStringPos - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPosAt
- GpiPop
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryDefCharBox
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix

GpiCharStringPos - Graphic Elements and Orders

Element Type: **ETYPE_GCCHSTE**

Order: Character String Extended at Current Position

GpiCharStringPos - Example Code

This example uses GpiCharStringPos to display "13 characters", starting at position 10,10 and clipped to a 100x100 rectangle in the lower left corner.

```
#define INCL_GPIPRIMITIVES      /* GPI Primitive functions      */
#include <os2.h>
```

```
LONG  lHits;            /* correlation/error indicator          */
HPS   hps;              /* Presentation-space handle            */
POINTL pptlStart = {10L,10L};
                        /* Starting position */
RECTL  prclRect = {0L,0L,100L,100L};
                        /* Rectangle structure                  */
ULONG  flOptions;       /* Formatting options                   */
LONG  lCount;           /* Number of bytes in the string        */
char  pchString[25];    /* Characters to be drawn               */

GpiMove(hps, &pptlStart);

flOptions = CHS_CLIP; /* clip text to rectangle */
lCount = 13;
strcpy(pchString,"13 characters");

/* draw the string */
lHits = GpiCharStringPos(hps, &prclRect, flOptions, lCount,
                         pchString, NULL);
```

GpiCharStringPos - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiCharStringPosAt

GpiCharStringPosAt - Syntax

This function draws a character string starting at a specified position, with formatting options.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlStart;  /*  Starting position. */
PRECTL     prclRect;   /*  Rectangle structure. */
ULONG      flOptions;  /*  Formatting options. */
LONG       lCount;     /*  Number of bytes in the string. */
PCH        pchString;  /*  Character string. */
PLONG      alAdx;      /*  Increment values. */
LONG       lHits;      /*  Correlation and error indicators. */

lHits = GpiCharStringPosAt(hps, pptlStart,
          prclRect, flOptions, lCount, pchString,
```

```
alAdx);
```

GpiCharStringPosAt Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCharStringPosAt Parameter - pptlStart

pptlStart (PPOINTL) - input
Starting position.

GpiCharStringPosAt Parameter - prclRect

prclRect (PRECTL) - input
Rectangle structure.

Defines, in world coordinates, the two corners of the rectangle that defines the background of the characters. It is ignored unless CHS_OPAQUE or CHS_CLIP is selected.

GpiCharStringPosAt Parameter - flOptions

flOptions (ULONG) - input
Formatting options.

Option flags that can be used in combination:

CHS_OPAQUE
: Background of characters is defined by the rectangle specified by *prclRect*. The rectangle is to be shaded (with background color and overpaint) before drawing.

CHS_VECTOR
: Increments vector (*alAdx*) is supplied. If 0, *alAdx* is ignored.

CHS_LEAVEPOS
: If set, current position is unchanged by this function. If not set, current position is moved to the position at which the next character would have been drawn, had there been one.

CHS_CLIP
: Clip the string to the rectangle.

CHS_UNDERSCORE
: Underscore the characters. See FATTR_SEL_UNDERSCORE in the FATTRS datatype.

CHS_STRIKEOUT
: Overstrike the characters. See FATTR_SEL_STRIKEOUT in the FATTRS datatype.

Other bits are reserved and must be zero.

GpiCharStringPosAt Parameter - lCount

lCount (LONG) - input
Number of bytes in the string.

Must be greater than 0 and less or equal to 512.

GpiCharStringPosAt Parameter - pchString

pchString (PCH) - input
Character string.

This parameter does not need to be null terminated.

GpiCharStringPosAt Parameter - alAdx

alAdx (PLONG) - input
Increment values.

Vector of increment values, in world coordinates. Any negative values are treated as if they were zero.

GpiCharStringPosAt Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiCharStringPosAt - Parameters

hps (HPS) - input
Presentation-space handle.

pptlStart (PPOINTL) - input
Starting position.

prclRect (PRECTL) - input
Rectangle structure.

Defines, in world coordinates, the two corners of the rectangle that defines the background of the characters. It is ignored unless CHS_OPAQUE or CHS_CLIP is selected.

flOptions (ULONG) - input
Formatting options.

Option flags that can be used in combination:

CHS_OPAQUE
Background of characters is defined by the rectangle specified by *prclRect*. The rectangle is to be shaded (with background color and overpaint) before drawing.

CHS_VECTOR
Increments vector (*alAdx*) is supplied. If 0, *alAdx* is ignored.

CHS_LEAVEPOS
If set, current position is unchanged by this function. If not set, current position is moved to the position at which the next character would have been drawn, had there been one.

CHS_CLIP
Clip the string to the rectangle.

CHS_UNDERSCORE
Underscore the characters. See FATTR_SEL_UNDERSCORE in the FATTRS datatype.

CHS_STRIKEOUT
Overstrike the characters. See FATTR_SEL_STRIKEOUT in the FATTRS datatype.

Other bits are reserved and must be zero.

lCount (LONG) - input
Number of bytes in the string.

Must be greater than 0 and less or equal to 512.

pchString (PCH) - input
Character string.

This parameter does not need to be null terminated.

alAdx (PLONG) - input
Increment values.

Vector of increment values, in world coordinates. Any negative values are treated as if they were zero.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiCharStringPosAt - Remarks

A vector of increments can be specified, allowing control over the position of each character after the first. This vector consists of distances measured in world coordinates (along the baseline for left-to-right and right-to-left character directions, and along the shearline for top-to-bottom and bottom-to-top character directions). Increment *i* is the distance of the reference point (for example, lower left corner) of character *i+1* from the reference point of character *i*. The last increment may be needed to update the current position.

These increments, if specified, set the widths of each character.

A further option allows a rectangle to be specified that can be used as the background of the string instead of the normal background. This rectangle is painted using the current character background color and an overpaint mix (unless this is in a dynamic segment, when leave-alone is used). Both corners of the rectangle are specified, so that the rectangle is positioned independently of current position. Points on the borders of the rectangle are considered to be included within the rectangle.

Clipping of the string to the rectangle is also allowed. This is independent of whether the rectangle is actually drawn.

Current position can be updated to the point at which the next character would have been drawn, had there been one, or it can be left at the start of the string.

GpiCharStringPosAt - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_POS_OPTIONS (0x204E)
: An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
: An invalid rectangle parameter was specified.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_FONT_AND_MODE_MISMATCH (0x202D)
: An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.

GpiCharStringPosAt - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiPop
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryDefCharBox
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetCharAngle

- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix

GpiCharStringPosAt - Graphic Elements and Orders

Element Type: **ETYPE_GCHSTE**

Order: Character String Extended at Given Position

GpiCharStringPosAt - Example Code

This example uses GpiCharStringPosAt to display "13 Characters", starting at position (10,10) and clipped to a 100x100 rectangle in the lower left corner.

```
#define INCL_GPIPRIMITIVES      /* GPI Primitive functions      */
#include <os2.h>

LONG  lHits;            /* correlation/error indicator          */
HPS   hps;              /* Presentation-space handle            */
POINTL  pptlStart = {10L,10L};
                        /* Starting position */
RECTL  rclRect = {0L,0L,100L,100L};
                        /* Rectangle structure                  */
ULONG  flOptions;       /* Formatting options                   */
LONG  lCount;           /* Number of bytes in the string        */
char  pchString[14];    /* Characters to be drawn               */

flOptions = CHS_CLIP; /* clip text to rectangle */
lCount = 13;
strcpy(pchString,"13 characters");

lHits = GpiCharStringPosAt(hps, &pptlStart, &rclRect, flOptions,
                           lCount, pchString, NULL);
```

GpiCharStringPosAt - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders

GpiCloseFigure

GpiCloseFigure - Syntax

This function closes a figure within a path specification.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiCloseFigure(hps);
```

GpiCloseFigure Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCloseFigure Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiCloseFigure - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiCloseFigure - Remarks

The current figure is closed by a line drawn to the start point of the figure.

This function need not be used if the path is to be filled (see GpiFillPath), or used as a clip path (see GpiSetClipPath), as any figures in the path that have not been closed are automatically closed at that time. It should be used, however, for any closed figures within paths that are subsequently to be stroked by GpiModifyPath or GpiStrokePath.

This function must not be used outside a path specification. In particular, it must not be used within an area.

GpiCloseFigure - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiCloseFigure - Related Functions

Prerequisite Functions

- GpiBeginPath

Related Functions

- GpiEndPath
- GpiModifyPath
- GpiStrokePath

GpiCloseFigure - Graphic Elements and Orders

Element Type: **OCODE_GCFIG**

Order: Close Figure

GpiCloseFigure - Example Code

This example uses the GpiCloseFigure function to close a triangle drawn in a path bracket. The triangle starts at (0,0), and as the current position just before the GpiCloseFigure is (200,0), the function closes the triangle by drawing a line from (200,0) to (0,0).

```
#define INCL_GPIPATHS           /* GPI Path functions           */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 };
POINTL ptlPoints[] = { 100, 100, 200, 0 };

GpiBeginPath(hps, 1L);                /* start the path bracket */
GpiMove(hps, &ptlStart);              /* move to starting point */
GpiPolyLine(hps, 2L, ptlPoints);      /* draw two sides         */
GpiCloseFigure(hps);                  /* close the triangle     */
GpiEndPath(hps);                      /* end the path bracket   */
```

GpiCloseFigure - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiCloseSegment

GpiCloseSegment - Syntax

This function closes the current segment.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiCloseSegment(hps);
```

GpiCloseSegment Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCloseSegment Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiCloseSegment - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiCloseSegment - Remarks

Closing a segment does not delete the segment or affect the graphics primitives that are drawn.

Any attributes that have been preserved (see the AM_PRESERVE option of GpiSetAttrMode) are popped (restored) when the GpiCloseSegment function is issued in **draw** or **draw-and-retain** modes, and at the end of the segment when the segment is subsequently drawn in **draw-and-retain** or **retain** modes (see GpiSetDrawingMode).

If an area or path is open when a segment is closed, the area or path is terminated. When the drawing mode is **draw** or **draw-and-retain**, a warning is given, but the close processing continues. No warning is given for **retain** mode. If a retained segment with an open area or path is drawn, an error occurs.

If an element bracket is open when a segment is closed, the element bracket is first closed automatically.

If this function is followed by primitives or attributes, without first opening a segment, the following may or may not have been reset to their default values:

- Current attribute values and arc parameters
- Current tag
- Current model transform
- Current position
- Current clip path and viewing limits.

Any such quantity can be assumed to contain its default value only if it is known either that it has not been changed from the default, or that last time it was changed, it was set to its default value. An application should not be written to depend on the values of these quantities immediately after GpiCloseSegment.

Subsequent primitives, not preceded by an GpiOpenSegment function, are not retained, irrespective of the current drawing mode.

The current viewing transform, however, is guaranteed to be reset to unity for primitives outside segments.

GpiCloseSegment - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_SEG (0x20E3)
An attempt was made to end a segment using GpiCloseSegment while not in a segment bracket.

PMERR_PATH_INCOMPLETE (0x20EC)
An attempt was made to open or close a segment either directly or during segment drawing, or to issue GpiAssociate while there is an open path bracket.

PMERR_AREA_INCOMPLETE (0x2005)
One of the following has occurred:
o A segment has been opened, closed, or drawn.
o GpiAssociate was issued while an area bracket was open.
o A drawn segment has opened an area bracket and ended without closing it.

GpiCloseSegment - Related Functions

Prerequisite Functions

- GpiOpenSegment

Related Functions

- GpiCallSegmentMatrix
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiQueryInitialSegmentAttrs

- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

GpiCloseSegment - Example Code

This example uses the GpiCloseSegment function to close a segment. The GpiOpenSegment opens the segment; GpiMove and GpiPolyLine draw a triangle.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiOpenSegment(hps, 1L);            /* open the segment          */
GpiMove(hps, &ptlStart);            /* move to start point (0,0) */
GpiPolyLine(hps, 3L, ptlTriangle);  /* draw triangle             */
GpiCloseSegment(hps);               /* close the segment         */
```

GpiCloseSegment - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCombineRegion

GpiCombineRegion - Syntax

This function combines two regions.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;          /*  Presentation-space handle. */
```

```
HRGN    hrgnDest;     /*  Handle of destination. */
HRGN    hrgnSrc1;     /*  Handle of first source region. */
HRGN    hrgnSrc2;     /*  Handle of second source region. */
LONG    lMode;        /*  Method of combination. */
LONG    lComplexity;  /*  Complexity of resulting region and error indicators. */

lComplexity = GpiCombineRegion(hps, hrgnDest,
                hrgnSrc1, hrgnSrc2, lMode);
```

GpiCombineRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

The regions must be owned by the device identified by the currently associated device context.

GpiCombineRegion Parameter - hrgnDest

hrgnDest (HRGN) - input
Handle of destination.

GpiCombineRegion Parameter - hrgnSrc1

hrgnSrc1 (HRGN) - input
Handle of first source region.

GpiCombineRegion Parameter - hrgnSrc2

hrgnSrc2 (HRGN) - input
Handle of second source region.

GpiCombineRegion Parameter - lMode

lMode (LONG) - input
Method of combination.

CRGN_OR
: Union of *hrgnSrc1* and *hrgnSrc2*

CRGN_COPY
: *hrgnSrc1* only (*hrgnSrc2* ignored)

CRGN_XOR
: Symmetric difference of *hrgnSrc1* and *hrgnSrc2*

CRGN_AND
: Intersection of *hrgnSrc1* and *hrgnSrc2*

CRGN_DIFF
: *hrgnSrc1* *and not* (*hrgnSrc2*).

GpiCombineRegion Return Value - lComplexity

lComplexity (LONG) - returns
Complexity of resulting region and error indicators.

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiCombineRegion - Parameters

hps (HPS) - input
Presentation-space handle.

The regions must be owned by the device identified by the currently associated device context.

hrgnDest (HRGN) - input
Handle of destination.

hrgnSrc1 (HRGN) - input
Handle of first source region.

hrgnSrc2 (HRGN) - input
Handle of second source region.

lMode (LONG) - input
Method of combination.

CRGN_OR
: Union of *hrgnSrc1* and *hrgnSrc2*

CRGN_COPY
: *hrgnSrc1* only (*hrgnSrc2* ignored)

CRGN_XOR
: Symmetric difference of *hrgnSrc1* and *hrgnSrc2*

CRGN_AND
: Intersection of *hrgnSrc1* and *hrgnSrc2*

CRGN_DIFF
: *hrgnSrc1* *and not* (*hrgnSrc2*).

lComplexity (LONG) - returns
Complexity of resulting region and error indicators.

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiCombineRegion - Remarks

Source and destination regions must all be of the same device class. The destination region can be one of the source regions.

An error is raised if any of the specified regions are currently selected as the clip region (by GpiSetClipRegion).

GpiCombineRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
: An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
: An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_INV_REGION_MIX_MODE (0x20BF)
: An invalid mode parameter was specified with GpiCombineRegion.

PMERR_HRGN_BUSY (0x2034)
: An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiCombineRegion - Related Functions

Related Functions

- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

GpiCombineRegion - Example Code

This example uses the GpiCombineRegion function to create a complex region consisting of everything in two rectangles except where they overlap.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>

HPS hps;                /* presentation space handle            */
HRGN hrgn1, hrgn2, hrgn3;
RECTL rclRect1 = { 0, 0, 100, 100 };
RECTL rclRect2 = { 50, 50, 200, 200 };

/* create first region  */
hrgn1 = GpiCreateRegion(hps, 1L, &rclRect1);
/* create second region */
hrgn2 = GpiCreateRegion(hps, 1L, &rclRect2);
/* create empty region  */
hrgn3 = GpiCreateRegion(hps, 0L, NULL);

/* Combine first and second regions, replacing the empty region. */

GpiCombineRegion(hps, hrgn3, hrgn1, hrgn2, CRGN_XOR);
```

GpiCombineRegion - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiComment

GpiComment - Syntax

This function adds a comment to the current segment.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;      /*  Presentation-space handle. */
LONG     lLength;  /*  Data length. */
PBYTE    pbData;   /*  Comment string. */
BOOL     rc;       /*  Success indicator. */
```

```
rc = GpiComment(hps, lLength, pbData);
```

GpiComment Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiComment Parameter - lLength

lLength (LONG) - input
Data length.

The length of *pbData* in bytes. *lLength* must >= 0 and <= 255.

GpiComment Parameter - pbData

pbData (PBYTE) - input
Comment string.

No conversion of any kind is performed on the data.

GpiComment Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiComment - Parameters

hps (HPS) - input
Presentation-space handle.

lLength (LONG) - input
Data length.

The length of *pbData* in bytes. *lLength* must >= 0 and <= 255.

pbData (PBYTE) - input
Comment string.

No conversion of any kind is performed on the data.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiComment - Remarks

An application can use this function to store some data of its own in the segment if the drawing mode (see GpiSetDrawingMode) is set to **retain** or **draw-and-retain**. It has no effect on drawing. The data can subsequently be retrieved by the application using GpiQueryElement or GpiGetData.

GpiComment - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

GpiComment - Graphic Elements and Orders

Element Type: **OCODE_GCOMT**

Order: Comment

GpiComment - Example Code

This example uses the GpiComment function to comment the contents of a segment.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiOpenSegment(hps, 0L);                    /* open the segment  */
GpiComment(hps, 18L, "Start point (0, 0)");
GpiMove(hps, &ptlStart);
GpiComment(hps, 13L, "Draw triangle");
GpiPolyLine(hps, 3L, ptlTriangle);
GpiCloseSegment(hps);                       /* close the segment */
```

GpiComment - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Graphic Elements and Orders
Glossary

GpiConvert

GpiConvert - Syntax

This function converts an array of coordinate pairs from one coordinate space to another.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lSrc;        /*  Source coordinate space. */
LONG       lTarg;       /*  Target coordinate space. */
LONG       lCount;      /*  Number of coordinate pairs in aptlPoints. Must be greater or equal to 0. */
PPOINTL    aptlPoints;  /*  Array of coordinate pair structures. */
BOOL       rc;          /*  Success indicator. */

rc = GpiConvert(hps, lSrc, lTarg, lCount,
       aptlPoints);
```

GpiConvert Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiConvert Parameter - lSrc

lSrc (LONG) - input
Source coordinate space.

CVTC_WORLD
World coordinates
CVTC_MODEL
Model space
CVTC_DEFAULTPAGE
Page space before default viewing transform
CVTC_PAGE
Page space after default viewing transform
CVTC_DEVICE
Device space.

GpiConvert Parameter - lTarg

lTarg (LONG) - input
Target coordinate space.

CVTC_WORLD
World coordinates
CVTC_MODEL
Model space
CVTC_DEFAULTPAGE
Page space before default viewing transform
CVTC_PAGE
Page space after default viewing transform
CVTC_DEVICE
Device space.

GpiConvert Parameter - lCount

lCount (LONG) - input
Number of coordinate pairs in *aptlPoints* . Must be greater or equal to 0.

GpiConvert Parameter - aptlPoints

aptlPoints (PPOINTL) - in/out
Array of coordinate pair structures.

GpiConvert Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiConvert - Parameters

hps (HPS) - input
Presentation-space handle.

lSrc (LONG) - input
Source coordinate space.

CVTC_WORLD
World coordinates
CVTC_MODEL
Model space
CVTC_DEFAULTPAGE
Page space before default viewing transform
CVTC_PAGE
Page space after default viewing transform
CVTC_DEVICE
Device space.

lTarg (LONG) - input
Target coordinate space.

CVTC_WORLD
World coordinates
CVTC_MODEL
Model space
CVTC_DEFAULTPAGE
Page space before default viewing transform
CVTC_PAGE
Page space after default viewing transform
CVTC_DEVICE
Device space.

lCount (LONG) - input
Number of coordinate pairs in *aptlPoints* . Must be greater or equal to 0.

aptlPoints (PPOINTL) - in/out
Array of coordinate pair structures.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE

Error occurred.

GpiConvert - Remarks

This function replaces each coordinate pair in *aptlPoints* with the converted values.

Conversions involving either world coordinates or model space should not be performed if the drawing mode (see GpiSetDrawingMode) is **retain**.

GpiConvert - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COORD_SPACE (0x205A)
An invalid source or target coordinate space parameter was specified with GpiConvert.

PMERR_COORDINATE_OVERFLOW (0x2014)
An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiConvert - Related Functions

Related Functions

- GpiCreatePS
- GpiSetDefaultViewMatrix
- GpiSetModelTransformMatrix
- GpiSetPageViewport
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix

GpiConvert - Example Code

This example uses the GpiConvert function to convert the coordinates of the mouse pointer to the corresponding coordinates in world space. The system passes mouse coordinates to a window procedure in the WM_MOUSEMOVE message. The coordinates are device coordinates. After the coordinates are converted, the GpiMove uses them to move to a new location in world space.

```
#define INCL_GPITRANSFORMS      /* GPI Transform functions      */
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

MPARAM mp1;
HPS    hps;
POINTL ptl;

case WM_MOUSEMOVE:
    ptl.x = (LONG) SHORT1FROMMP(mp1);
    ptl.y = (LONG) SHORT2FROMMP(mp1);
    GpiConvert(hps, CVTC_DEVICE, CVTC_WORLD, 1L, &ptl);
    GpiMove(hps, &ptl);
```

GpiConvert - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiConvertWithMatrix

GpiConvertWithMatrix - Syntax

This function converts an array of (x,y) coordinate pairs from one coordinate space to another, using the supplied transform matrix.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
LONG         lCountp;      /*  Point count. */
PPOINTL      aptlPoints;   /*  Array of (x,y) coordinate pair structures. */
LONG         lCount;       /*  Number of elements. */
PMATRIXLF    pmatlfArray;  /*  Instance transform matrix. */
BOOL         rc;           /*  Success indicator. */

rc = GpiConvertWithMatrix(hps, lCountp, aptlPoints,
       lCount, pmatlfArray);
```

GpiConvertWithMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiConvertWithMatrix Parameter - lCountp

lCountp (LONG) - input
Point count.

Number of coordinate pairs in *aptlPoints* .

GpiConvertWithMatrix Parameter - aptlPoints

aptlPoints (PPOINTL) - in/out
Array of (x,y) coordinate pair structures.

GpiConvertWithMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements of *pmatlfArray* to be examined, starting from the beginning of the structure. If *lCount* is less or equal to 9, remaining elements default to the corresponding elements of the identity matrix. If *lCount* = 0, the identity matrix is used.

GpiConvertWithMatrix Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - input
Instance transform matrix.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

GpiConvertWithMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiConvertWithMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lCountp (LONG) - input
Point count.

Number of coordinate pairs in *aptlPoints* .

aptlPoints (PPOINTL) - in/out
Array of (x,y) coordinate pair structures.

lCount (LONG) - input
Number of elements.

The number of elements of *pmatlfArray* to be examined, starting from the beginning of the structure. If *lCount* is less or equal to 9, remaining elements default to the corresponding elements of the identity matrix. If *lCount* = 0, the identity matrix is used.

pmatlfArray (PMATRIXLF) - input
Instance transform matrix.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiConvertWithMatrix - Remarks

The array contains x1, y1, x2, y2,.... The input coordinates are replaced by the converted coordinates.

Only the supplied transform matrix is used, all other current transforms are ignored by this function.

The transform is specified as a one-dimensional array of elements, being the first *lCount* elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

```
   Matrix                  Array


  a  b  0
  c  d  0            (a,b,0,c,d,0,e,f,1)
  e  f  1
```

A point with coordinates (x,y) is transformed to the point

```
(a*x + c*y + e, b*x + d*y + f)
```

GpiConvertWithMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW (0x2014)
: An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiConvertWithMatrix - Example Code

This example uses GpiConvertWithMatrix to convert two coordinate pairs to another coordinate space defined by the supplied matrix, which has only the first transform element defined.

```
#define INCL_GPITRANSFORMS      /* GPI Transform functions      */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS  hps;               /* Presentation-space handle            */
LONG  lCountp;          /* Point count                          */
POINTL  aptlPoints[2] = {{0L,0L},{1L,1L}};
                        /* Array of (x,y) coordinate pair
                           structures                           */
LONG  lCount;           /* Number of elements                   */
MATRIXLF  pmatlfArray;  /* Instance transform matrix            */

lCount = 1; /* examine only first element of transform matrix */

pmatlfArray.fxM11 = 2; /* set first element of transform matrix */

fSuccess = GpiConvertWithMatrix(hps, lCountp, aptlPoints,
                                lCount, &pmatlfArray);
```

GpiConvertWithMatrix - Topics

Select an item:
Syntax
Parameters

Returns
Errors
Remarks
Example Code
Glossary

GpiCopyMetaFile

GpiCopyMetaFile - Syntax

This function creates a new metafile and copies the contents of an existing loaded metafile into it.

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HMF    hmf;     /*  Source metafile handle. */
HMF    hmfNew;  /*  New metafile handle and error indicators. */

hmfNew = GpiCopyMetaFile(hmf);
```

GpiCopyMetaFile Parameter - hmf

hmf (HMF) - input
Source metafile handle.

GpiCopyMetaFile Return Value - hmfNew

hmfNew (HMF) - returns
New metafile handle and error indicators.

<>0
New metafile handle
GPI_ERROR
Error.

GpiCopyMetaFile - Parameters

hmf (HMF) - input
Source metafile handle.

hmfNew (HMF) - returns
New metafile handle and error indicators.

<>0
New metafile handle
GPI_ERROR
Error.

GpiCopyMetaFile - Remarks

The source metafile must already be loaded or generated. It is identified by a metafile handle. The new metafile is identified by a handle that is returned by this function, so it may be used, for example, by GpiPlayMetaFile.

The new metafile is owned by the process from which this function is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system.

The maximum number of metafiles allowed for a given process depends on memory construction and the application.

GpiCopyMetaFile - Errors

Possible returns from WinGetLastError

PMERR_INV_HMF (0x207E)
An invalid metafile handle was specified.

PMERR_METAFILE_IN_USE (0x20D9)
An attempt has been made to access a metafile that is in use by another thread.

PMERR_TOO_MANY_METAFILES_IN_USE (0x2106)
The maximum number of metafiles allowed for a given process was exceeded.

GpiCopyMetaFile - Related Functions

Related Functions

- GpiDeleteMetaFile
- GpiLoadMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile
- GpiSetMetaFileBits

GpiCopyMetaFile - Example Code

This example uses the GpiCopyMetaFile function to make a copy of the metafile loaded using the GpiLoadMetaFile function.

```
#define INCL_GPIMETAFILES       /* Metafile functions           */
#include <os2.h>

HAB hab;                /* anchor block handle                  */
HMF hmf, hmf2;          /* metafile handle                      */

/* loads metafile from disk */
hmf = GpiLoadMetaFile(hab, "sample.met");
    .
    .
    .
hmf2 = GpiCopyMetaFile(hmf);          /* copy the metafile       */
```

GpiCopyMetaFile - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCorrelateChain

GpiCorrelateChain - Syntax

This function performs a correlate operation on the retained segment chain. It returns data for each tagged primitive that intersects the current aperture, as set by GpiSetPickApertureSize.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
LONG       lType;      /*  Segment type. */
PPOINTL    pptlPick;   /*  Pick position. */
LONG       lMaxHits;   /*  Maximum hits. */
LONG       lMaxDepth;  /*  Number of pairs. */
PLONG      pl2;        /*  Segment identifiers and tags. */
LONG       lNumHits;   /*  Number of hits and error indicators. */

lNumHits = GpiCorrelateChain(hps, lType, pptlPick,
             lMaxHits, lMaxDepth, pl2);
```

GpiCorrelateChain Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCorrelateChain Parameter - lType

lType (LONG) - input
Segment type.

Type of segment on which correlation is to be performed:

PICKSEL_VISIBLE
Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL
All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

GpiCorrelateChain Parameter - pptlPick

pptlPick (PPOINTL) - input
Pick position.

The position of the center of the pick aperture, in presentation page units.

GpiCorrelateChain Parameter - lMaxHits

lMaxHits (LONG) - input
Maximum hits.

Maximum number of hits that can be returned in the *pl2* parameter. It must be greater or equal to 0.

GpiCorrelateChain Parameter - lMaxDepth

lMaxDepth (LONG) - input
Number of pairs.

Number of segment and tag pairs to be returned by each hit. It must be greater or equal to 0.

GpiCorrelateChain Parameter - pl2

pl2 (PLONG) - output
Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *lMaxDepth* segment identifiers and tag pairs is returned.

GpiCorrelateChain Return Value - lNumHits

lNumHits (LONG) - returns
Number of hits and error indicators.

>=0
Number of hits that occurred
GPI_ALTERROR
Error.

GpiCorrelateChain - Parameters

hps (HPS) - input
Presentation-space handle.

lType (LONG) - input
Segment type.

Type of segment on which correlation is to be performed:

PICKSEL_VISIBLE
Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL
All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

pptlPick (PPOINTL) - input
Pick position.

The position of the center of the pick aperture, in presentation page units.

lMaxHits (LONG) - input
Maximum hits.

Maximum number of hits that can be returned in the *pl2* parameter. It must be greater or equal to 0.

lMaxDepth (LONG) - input
Number of pairs.

Number of segment and tag pairs to be returned by each hit. It must be greater or equal to 0.

pl2 (PLONG) - output
Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *lMaxDepth* segment

identifiers and tag pairs is returned.

lNumHits (LONG) - returns
Number of hits and error indicators.

>=0
Number of hits that occurred
GPI_ALTERROR
Error.

GpiCorrelateChain - Remarks

The data returned for each "hit" (or correlation) consists of a set of segment and tag pairs, starting with the correlated one and followed by the one that called that segment. This is repeated until either the root segment is reached or *lMaxDepth* segment and tag pairs are returned.

Only primitives with a nonzero tag in segments with a nonzero identifier are correlated using this function. Primitives in segments called (to any depth in the hierarchy) from an unnamed segment are not eligible for correlation.

The depth value specifies the number of sets of segment and tag pairs to be returned for each hit. If the root segment is reached before *lMaxDepth* values, the remaining values are set to zero. If more than *lMaxDepth* values are available, only that number is returned.

The number of hits that occurred is returned in *lNumHits*.

A "hit" is an instance of a segment identifier and tag pair for which the primitives lie completely or partially within the specified aperture. Two different primitives in the same segment might have the same tag, and would therefore produce the same hit. This is counted as a single hit; the hit is recorded only once in the *pl2* parameter returned. The *lNumHits* parameter, therefore, returns this distinct number of hits. Hits are returned in the reverse order of their occurrence.

pl2 is set to the hits that are found, up to the maximum defined in the *lMaxHits* parameter. Corresponding pairs of elements form the "hit" pairs. The number returned by the function, therefore, contains the number of sets of *lMaxDepth* pairs set if the *lMaxHits* parameter is greater than the number of hits detected. The number of elements set in the *pl2* parameter is twice the number returned by the function (subject to a maximum of *lMaxHits*) multiplied by the *lMaxDepth*.

If the *lNumHits* value returned by the function is greater than that specified in *lMaxHits*, more hits occurred than could be returned. If all hits are important, specify an array that is large enough to contain the maximum number of sets of hits that are expected.

The draw controls (see GpiSetDrawControl) are ignored by this function.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the chain. This can be done by either ensuring that the first segment to be correlated does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiCorrelateChain. The latter method also resets the clip path to no clipping.

If this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

Examples

```
Start segment 1
  Tag 10
  Call 2
End segment 1

Start segment 2
  Tag 20                        Pick Aperture
  Call 3
  Tag 21
  ......                         Hit 1
  ......
End segment 2

Start segment 3
  Tag 30
  ......                         Hit 2
  ......
End segment 3
```

For *lMaxHits* = 1 at *lMaxDepth* = 2:

```
segment        tag
   2            21
   1            10
```

Returned *lNumHits* = 2.

For *lMaxHits* = 2 at *lMaxDepth* = 4:

```
segment        tag
   2            21       hit1.1
   1            10       hit1.2
   0            0        hit1.3
   0            0        hit1.4
   3            30       hit2.1
   2            20       hit2.2
   1            10       hit2.3
   0            0        hit2.4
```

Returned *lNumHits* = 2.

GpiCorrelateChain - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_MAX_HITS (0x209C)
An invalid maxhits parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

PMERR_INV_CORRELATE_DEPTH (0x205C)
An invalid maxdepth parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_CORRELATE_TYPE (0x205D)
An invalid type parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

GpiCorrelateChain - Related Functions

Related Functions

- GpiCorrelateFrom
- GpiCorrelateSegment
- GpiSetDrawControl

- GpiSetPickAperturePosition
- GpiSetPickApertureSize

GpiCorrelateChain - Example Code

This example uses GpiCorrelateChain to correlate, using an aperture of default size and centered at (200,200), on visible and detectable segments and requests one intersection (or hit) and one segment/tag pair for that hit to be returned. The segments will have been previously defined and created using GpiSetInitialSegmentAttrs and GpiOpenSegment/GpiCloseSegment.

```
#define INCL_GPICORRELATION     /* GPI Correlation functions    */
#include <os2.h>

BOOL     fSuccess;      /* success indicator                   */
SIZEL    psizlSize={0L,0L}; /* size of pick aperture           */
LONG     lNumHits;      /* number of hits or error             */
HPS      hps;           /* Presentation-space handle           */
POINTL   pptlPick = {200L,200L};
                        /* Pick (center of aperture) position  */
LONG     lMaxHits;      /* Maximum hits to be returned         */
LONG     lMaxDepth;     /* Number of pairs to be returned      */
LONG     alSegTag;      /* Segment identifiers and tags        */

fSuccess = GpiSetPickAperturePosition(hps, &pptlPick);

/* set aperture size (use default) */
fSuccess = GpiSetPickApertureSize(hps, PICKAP_DEFAULT, &psizlSize);

/* return only one hit */
lMaxHits = 1L;

/* return only one segment/tag pair per hit */
lMaxDepth = 1L;

/* correlate on visible, detectable segment chains */
lNumHits = GpiCorrelateChain(hps, PICKSEL_VISIBLE, &pptlPick, lMaxHits,
                                lMaxDepth, &alSegTag);
```

GpiCorrelateChain - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCorrelateFrom

GpiCorrelateFrom - Syntax

This function performs a correlate operation on a section of the retained segment chain.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;            /*  Presentation-space handle. */
LONG       lFirstSegment;  /*  Specifies the first segment to be correlated. */
LONG       lLastSegment;   /*  Specifies the last segment to be correlated. */
LONG       lType;          /*  Type of segments on which correlation is to be performed. */
PPOINTL    pptlPick;       /*  Pick position. */
LONG       lMaxHits;       /*  Maximum hits. */
LONG       lMaxDepth;      /*  Number of pairs. */
PLONG      plSegTag;       /*  Segment identifiers and tags. */
LONG       lNumHits;       /*  Number of hits and error indicators. */

lNumHits = GpiCorrelateFrom(hps, lFirstSegment,
             lLastSegment, lType, pptlPick,
             lMaxHits, lMaxDepth, plSegTag);
```

GpiCorrelateFrom Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCorrelateFrom Parameter - lFirstSegment

lFirstSegment (LONG) - input
Specifies the first segment to be correlated.

It must be greater than 0.

GpiCorrelateFrom Parameter - lLastSegment

lLastSegment (LONG) - input
Specifies the last segment to be correlated.

It must be greater than 0.

GpiCorrelateFrom Parameter - lType

lType (LONG) - input
Type of segments on which correlation is to be performed.

PICKSEL_VISIBLE
Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL
All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

GpiCorrelateFrom Parameter - pptlPick

pptlPick (PPOINTL) - input
Pick position.

The position of the center of the pick aperture, in presentation page units.

GpiCorrelateFrom Parameter - lMaxHits

lMaxHits (LONG) - input
Maximum hits.

Maximum number of hits that can be returned in the *plSegTag* parameter. It must be greater than 0.

GpiCorrelateFrom Parameter - lMaxDepth

lMaxDepth (LONG) - input
Number of pairs.

Number of segment and tag pairs to be returned by each hit. It must be greater than 0.

GpiCorrelateFrom Parameter - plSegTag

plSegTag (PLONG) - output
Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *lMaxDepth* segment identifiers and tag pairs is returned.

GpiCorrelateFrom Return Value - lNumHits

lNumHits (LONG) - returns
Number of hits and error indicators.

>=0
Number of hits that occurred
GPI_ALTERROR
Error.

GpiCorrelateFrom - Parameters

hps (HPS) - input
Presentation-space handle.

lFirstSegment (LONG) - input
Specifies the first segment to be correlated.

It must be greater than 0.

lLastSegment (LONG) - input
Specifies the last segment to be correlated.

It must be greater than 0.

lType (LONG) - input
Type of segments on which correlation is to be performed.

PICKSEL_VISIBLE
Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL
All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

pptlPick (PPOINTL) - input
Pick position.

The position of the center of the pick aperture, in presentation page units.

lMaxHits (LONG) - input
Maximum hits.

Maximum number of hits that can be returned in the *plSegTag* parameter. It must be greater than 0.

lMaxDepth (LONG) - input
Number of pairs.

Number of segment and tag pairs to be returned by each hit. It must be greater than 0.

plSegTag (PLONG) - output
Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *lMaxDepth* segment identifiers and tag pairs is returned.

lNumHits (LONG) - returns
Number of hits and error indicators.

>=0
Number of hits that occurred

GPI_ALTERROR
Error.

GpiCorrelateFrom - Remarks

The correlation operation starts at the segment identified by *lFirstSegment* and includes chained and called segments up to, and including, the segment identified by *lLastSegment*.

Data is returned for each tagged primitive that intersects the pick aperture. The data returned for each "hit" (or correlation) consists of a set of segment and tag pairs, starting with the correlated one and followed by the one that called the segment. This is repeated until the root segment is reached or *lMaxDepth* values are returned.

Only primitives with a nonzero tag (see GpiSetTag) in segments with a nonzero identifier are correlated using this function. Primitives in segments called (to any depth in the hierarchy) from a segment 0 are not eligible for correlation.

The depth value specifies the number of sets of segment and tag pairs to be returned for each hit. If the root segment is reached before *lMaxDepth* values, the remaining values are set to zero. If more than *lMaxDepth* values are available, only that number is returned.

The number of hits that occurred is returned in *lNumHits*.

A "hit" is an instance of a segment identifier and tag pair for which the primitives lie completely or partially within the specified aperture. Two different primitives in the same segment might have the same tag, and would therefore produce the same hit. This is counted as a single hit; the hit is recorded only once in the *plSegTag* parameter returned. The *lNumHits* parameter, therefore, returns this distinct number of hits. Hits are returned in reverse order of their occurrence.

plSegTag is set to the hits that are found, up to the maximum defined in the *lMaxHits* parameter. Corresponding pairs of elements form the hit pairs. The number returned by the call therefore contains the number of sets of *lMaxDepth* pairs set if the *lMaxHits* parameter is greater than the number of hits detected. The number of elements set in the *plSegTag* parameter is twice the number returned by the function (subject to a maximum of *lMaxHits*) multiplied by the *lMaxDepth*.

If the *lNumHits* value returned by the function is greater than that specified in *lMaxHits*, more hits occurred than could be returned. If all hits are important, specify an array that is large enough to contain the maximum number of sets of hits that are expected.

The draw controls (see GpiSetDrawControl) are ignored by this call.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segments. This can be done either ensuring that the first segment to be correlated does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiCorrelateFrom. The latter method also resets the clip path to no clipping.

If this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

If *lFirstSegment* does not exist, or is not in the segment chain, an error is raised. If *lLastSegment* does not exist, or is not in the chain, or is chained before *lFirstSegment*, no error is raised and processing continues to the end of the chain.

GpiCorrelateFrom - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CORRELATE_TYPE (0x205D)
An invalid type parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_MAX_HITS (0x209C)
An invalid maxhits parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

PMERR_INV_CORRELATE_DEPTH (0x205C)
An invalid maxdepth parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_SEG_NOT_CHAINED (0x20FF)
An attempt was made to issue GpiDrawFrom, GpiCorrelateFrom or GpiQuerySegmentPriority for a segment that was not chained.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

GpiCorrelateFrom - Related Functions

Related Functions

- GpiCorrelateChain
- GpiCorrelateSegment
- GpiSetDrawControl
- GpiSetPickAperturePosition
- GpiSetPickApertureSize

GpiCorrelateFrom - Example Code

This example uses GpiCorrelateFrom to correlate, using an aperture of default size and centered at (200,200), on visible and detectable segments within the given chain of 2 segments. It requests one intersection (or hit) and one segment/tag pair for that hit to be returned. The segments will have been previously defined and created using GpiSetInitialSegmentAttrs and GpiOpenSegment/GpiCloseSegment.

```
#define INCL_GPICORRELATION     /* GPI Correlation functions    */
#include <os2.h>

BOOL     fSuccess;      /* success indicator                  */
SIZEL    psizlSize;     /* size of pick aperture              */
LONG     lNumHits;      /* number of hits or error            */
HPS      hps;           /* Presentation-space handle          */
LONG     lFirstSegment; /* Specifies the first segment to be
                           correlated                         */
LONG     lLastSegment;  /* Specifies the last segment to be
                           correlated                         */
POINTL   pptlPick = {200L,200L};
                        /* Pick (center of aperture) position */
LONG     lMaxHits;      /* Maximum hits to be returned        */
LONG     lMaxDepth;     /* Number of pairs to be returned     */
LONG     alSegTag;      /* Segment identifiers and tags       */

fSuccess = GpiSetPickAperturePosition(hps, &pptlPick);

/* set aperture size (use default) */
fSuccess = GpiSetPickApertureSize(hps, PICKAP_DEFAULT, &psizlSize);

/* define chain of two segments (1 and 2) */
lFirstSegment = 1;
lLastSegment = 2;

/* return only one hit */
lMaxHits = 1L;
```

```
/* return only one segment/tag pair per hit */
lMaxDepth = 1L;

/* correlate on visible, detectable segments */
lNumHits = GpiCorrelateFrom(hps, lFirstSegment, lLastSegment,
                            PICKSEL_VISIBLE, &pptlPick, lMaxHits,
                            lMaxDepth, &alSegTag);
```

GpiCorrelateFrom - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCorrelateSegment

GpiCorrelateSegment - Syntax

This function performs a correlate operation on a specified segment.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
LONG       lSegment;   /*  Identifier of the segment to be correlated. */
LONG       lType;      /*  Type of segments on which correlation is to be performed. */
PPOINTL    pptlPick;   /*  Pick position. */
LONG       lMaxHits;   /*  Maximum hits. */
LONG       lMaxDepth;  /*  Number of pairs. */
PLONG      alSegTag;   /*  Segment identifiers and tags. */
LONG       lNumHits;   /*  Number of hits and error indicators. */

lNumHits = GpiCorrelateSegment(hps, lSegment,
             lType, pptlPick, lMaxHits, lMaxDepth,
             alSegTag);
```

GpiCorrelateSegment Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCorrelateSegment Parameter - lSegment

lSegment (LONG) - input
Identifier of the segment to be correlated.

It must be greater than 0.

GpiCorrelateSegment Parameter - lType

lType (LONG) - input
Type of segments on which correlation is to be performed.

PICKSEL_VISIBLE
Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL
All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

GpiCorrelateSegment Parameter - pptlPick

pptlPick (PPOINTL) - input
Pick position.

The position of the center of the pick aperture, in presentation page units.

GpiCorrelateSegment Parameter - lMaxHits

lMaxHits (LONG) - input
Maximum hits.

The maximum number of hits that can be returned in the *alSegTag* parameter. It must be greater than 0.

GpiCorrelateSegment Parameter - lMaxDepth

lMaxDepth (LONG) - input
Number of pairs.

Number of segment/tag pairs to be returned by each hit. It must be greater than 0.

GpiCorrelateSegment Parameter - alSegTag

alSegTag (PLONG) - output
Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *lMaxDepth* segment identifiers and tag pairs is returned.

GpiCorrelateSegment Return Value - lNumHits

lNumHits (LONG) - returns
Number of hits and error indicators.

>=0
Number of hits that occurred
GPI_ALTERROR
Error.

GpiCorrelateSegment - Parameters

hps (HPS) - input
Presentation-space handle.

lSegment (LONG) - input
Identifier of the segment to be correlated.

It must be greater than 0.

lType (LONG) - input
Type of segments on which correlation is to be performed.

PICKSEL_VISIBLE
Only visible and detectable segments with nonzero identifiers are correlated.

PICKSEL_ALL
All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

pptlPick (PPOINTL) - input
Pick position.

The position of the center of the pick aperture, in presentation page units.

lMaxHits (LONG) - input

Maximum hits.

The maximum number of hits that can be returned in the *alSegTag* parameter. It must be greater than 0.

lMaxDepth (LONG) - input
Number of pairs.

Number of segment/tag pairs to be returned by each hit. It must be greater than 0.

alSegTag (PLONG) - output
Segment identifiers and tags.

An array consisting of segment identifiers and primitive tags in alternate elements. For each hit, a set of *lMaxDepth* segment identifiers and tag pairs is returned.

lNumHits (LONG) - returns
Number of hits and error indicators.

>=0
Number of hits that occurred
GPI_ALTERROR
Error.

GpiCorrelateSegment - Remarks

Data is returned for each tagged primitive that intersects the pick aperture. The data returned for each "hit" (or correlation) consists of a set of segment and tag pairs, starting with the correlated one and followed by the one that called that segment. This is repeated until the specified segment (which was not called by another segment) is reached, or *lMaxDepth* values are returned.

The specified segment identifier must be nonzero. Only primitives with a nonzero tag (see GpiSetTag) are correlated using this function.

The depth value specifies the number of sets of segment and tag pairs to be returned for each hit. If the specified segment is reached before *lMaxDepth* values, the remaining values are set to zero. If more than *lMaxDepth* values are available, only that number is returned.

The number of hits that occurred is returned in *lNumHits*.

A "hit" is an instance of a segment identifier and tag pair for which the primitives lie completely or partially within the specified aperture. Two different primitives in the same segment might have the same tag, and would therefore produce the same hit. This is counted as a single hit; the hit is recorded only once in the *alSegTag* parameter returned. The *lNumHits* parameter, therefore, returns this distinct number of hits. Hits are returned in reverse order of their occurrence.

alSegTag is set to the hits that are found, up to the maximum defined in the *lMaxHits* parameter. Corresponding pairs of elements form the hit pairs. The number returned by the function, therefore, contains the number of sets of *lMaxDepth* pairs set if the *lMaxHits* parameter is greater than the number of hits detected. The number of elements set in the *alSegTag* parameter is twice the number returned by the function (subject to a maximum of *lMaxHits*) multiplied by the *lMaxDepth*.

If the *lNumHits* value returned by the function is greater than that specified in *lMaxHits*, more hits occurred than could be returned. If all hits are important, specify an array that is large enough to contain the maximum number of sets of hits that are expected.

The draw controls (see GpiSetDrawControl) are ignored by this function. This function differs from the other GpiCorrelate... functions because the segment to be correlated need not be a chained segment.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values before processing the segment. This can be done either by ensuring that the segment to be correlated does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs) or by issuing GpiResetPS before the GpiCorrelateSegment. The latter method also resets the clip path to no clipping.

If this function is followed by primitives or attributes without first opening a segment, the processing is as described for GpiCloseSegment.

GpiCorrelateSegment - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CORRELATE_TYPE (0x205D)
An invalid type parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_MAX_HITS (0x209C)
An invalid maxhits parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

PMERR_INV_CORRELATE_DEPTH (0x205C)
An invalid maxdepth parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

GpiCorrelateSegment - Related Functions

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateChain
- GpiCorrelateFrom
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetDrawControl
- GpiSetInitialSegmentAttrs
- GpiSetPickAperturePosition
- GpiSetPickApertureSize
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

GpiCorrelateSegment - Example Code

This example uses GpiCorrelateSegment to correlate, using an aperture of default size and centered at (200,200), on a visible and detectable segment and requests one intersection (or hit) and one segment/tag pair for that hit to be returned. The segment will have been previously defined and created using GpiSetInitialSegmentAttrs and GpiOpenSegment/GpiCloseSegment.

```
#define INCL_GPICORRELATION     /* GPI Correlation functions    */
#include <os2.h>
```

```
BOOL     fSuccess;      /* success indicator                    */
SIZEL    psizlSize;     /* size of pick aperture                */
LONG     lNumHits;      /* number of hits or error              */
HPS      hps;           /* Presentation-space handle            */
LONG     lSegment;      /* segment to be correlated             */
LONG     lLastSegment;  /* Specifies the last segment to be
                           correlated                           */
POINTL   pptlPick = {200L,200L};
                        /* Pick (center of aperture) position   */
LONG     lMaxHits;      /* Maximum hits to be returned          */
LONG     lMaxDepth;     /* Number of pairs to be returned       */
LONG     alSegTag;      /* Segment identifiers and tags         */

fSuccess = GpiSetPickAperturePosition(hps, &pptlPick);

/* set aperture size (use default) */
fSuccess = GpiSetPickApertureSize(hps, PICKAP_DEFAULT, &psizlSize);

/* define segment */
lSegment = 1;

/* return only one hit */
lMaxHits = 1L;

/* return only one segment/tag pair per hit */
lMaxDepth = 1L;

/* correlate on visible, detectable segments */
lNumHits = GpiCorrelateSegment(hps, lSegment, PICKSEL_VISIBLE,
                               &pptlPick, lMaxHits, lMaxDepth,
                               &alSegTag);
```

GpiCorrelateSegment - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCreateBitmap

GpiCreateBitmap - Syntax

This function creates a bit map and returns the bit-map handle.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>
```

```
HPS                 hps;            /*  Presentation-space handle. */
PBITMAPINFOHEADER2  pbmpNew;        /*  Pointer to the bit-map information header. */
ULONG               flOptions;      /*  Options. */
PBYTE               pbInitData;     /*  Buffer address. */
PBITMAPINFO2        pbmiInfoTable;  /*  Pointer to the bit-map information table. */
HBITMAP             hbm;            /*  Bit-map handle and error indicators. */

hbm = GpiCreateBitmap(hps, pbmpNew, flOptions,
        pbInitData, pbmiInfoTable);
```

GpiCreateBitmap Parameter - hps

hps (HPS) - input
Presentation-space handle.

The associated device should, if possible, hold the bit map in its own memory. Where this is not possible, main memory is used and the bit map is held in a format compatible with the device.

GpiCreateBitmap Parameter - pbmpNew

pbmpNew (PBITMAPINFOHEADER2) - input
Pointer to the bit-map information header.

This structure defines the format of the bit map to be created.

GpiCreateBitmap Parameter - flOptions

flOptions (ULONG) - input
Options.

If the bit map is stored on a device, the *flOptions* parameter is passed to the device. Bits 16 through 31 can be used for special features known to be supported by the particular device driver.

The following option is supported:

CBM_INIT
Initialize the bit map with *pbInitData* .

GpiCreateBitmap Parameter - pbInitData

pbInitData (PBYTE) - input
Buffer address.

The address in application storage from which initialization data is to be copied, if CBM_INIT is set. This field must point to a doubleword aligned address to assure correct creation of the bit map by the device in device specific format.

GpiCreateBitmap Parameter - pbmiInfoTable

pbmiInfoTable (PBITMAPINFO2) - input
Pointer to the bit-map information table.

This defines the format of the data in *pbInitData* . It is ignored if CBM_INIT is not set.

GpiCreateBitmap Return Value - hbm

hbm (HBITMAP) - returns
Bit-map handle and error indicators.

<>0
New bit-map handle
GPI_ERROR
Error.

GpiCreateBitmap - Parameters

hps (HPS) - input
Presentation-space handle.

The associated device should, if possible, hold the bit map in its own memory. Where this is not possible, main memory is used and the bit map is held in a format compatible with the device.

pbmpNew (PBITMAPINFOHEADER2) - input
Pointer to the bit-map information header.

This structure defines the format of the bit map to be created.

flOptions (ULONG) - input
Options.

If the bit map is stored on a device, the *flOptions* parameter is passed to the device. Bits 16 through 31 can be used for special features known to be supported by the particular device driver.

The following option is supported:

CBM_INIT
Initialize the bit map with *pbInitData* .

pbInitData (PBYTE) - input
Buffer address.

The address in application storage from which initialization data is to be copied, if CBM_INIT is set. This field must point to a doubleword aligned address to assure correct creation of the bit map by the device in device specific format.

pbmiInfoTable (PBITMAPINFO2) - input

Pointer to the bit-map information table.

This defines the format of the data in *pbInitData* . It is ignored if CBM_INIT is not set.

hbm (HBITMAP) - returns
Bit-map handle and error indicators.

<>0
New bit-map handle
GPI_ERROR
Error.

GpiCreateBitmap - Remarks

On some devices it is possible to create the bit map in device memory. Even when this is not possible, a bit map always belongs to a particular device. The device is specified through the device context associated with the specified presentation space. The device context can be any device context that describes the physical device (such as any window device context for the screen).

There are a number of standard bit-map formats that should normally be adhered to. Other formats can be used if supported by the device.

A newly created bit map can be filled with data supplied by the application. This is useful where the bit map always contains, or always starts with, the same image, captured in the application. A bit-map information structure is also passed, which defines the format and color usage of the initialization data. It is assumed that enough data is passed to initialize the entire bit map.

Some bit-map functions, including those that draw into the bit map, require the bit map to be selected into a memory device context, using GpiSetBitmap. This is true whether device or main memory is used to hold the bit map.

The bit map is owned by the process from which this function is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system.

Some restrictions apply when using this function. Refer to "Format of Interchange Files" in the *Presentation Manager Programming Reference* for additional details.

GpiCreateBitmap - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_INFO_TABLE (0x208F)
An invalid bit-map info table was specified with a bit-map operation.

PMERR_INV_USAGE_PARM (0x20D1)
An invalid options parameter was specified with GpiCreateBitmap.

GpiCreateBitmap - Related Functions

Related Functions

- GpiBitBlt
- GpiDeleteBitmap

- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt

GpiCreateBitmap - Example Code

The following example loads a bit map resource from memory and uses the GpiCreateBitmap function to create the bit map. This is similar to using the GpiLoadBitmap function, except it gives the application the chance to modify the bit map image data before creating the bit map.

```
#define INCL_GPIBITMAPS         /* GPI bit map functions         */
#define INCL_DOSRESOURCES       /* Dos Resource functions       */
#include <os2.h>

HPS hps;                /* presentation space handle            */
                        /* address of bit map image data in
                           resource                             */
BITMAPINFOHEADER2 bmih; /* bit map info structure               */
HBITMAP hbm;            /* bit map handle                       */

memset (&bmih,0, sizeof(BITMAPINFOHEADER2));
bmih.cbFix        = sizeof(BITMAPINFOHEADER2);
bmih.cx           = 32;
bmih.cy           = 32;
bmih.cPlanes      = 1;
bmih.cBitCount = 32*32;

hbm = GpiCreateBitmap(hps, &bmih, 0L, NULL, NULL);
```

GpiCreateBitmap - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCreateLogColorTable

GpiCreateLogColorTable - Syntax

This function defines the entries of the logical color table.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
ULONG    flOptions;  /*  Options. */
LONG     lFormat;    /*  Format of entries in the table. */
LONG     lStart;     /*  Starting index. */
LONG     lCount;     /*  Count of elements in alTable. */
PLONG    alTable;    /*  Start of the application data area. */
BOOL     rc;         /*  Success indicator. */

rc = GpiCreateLogColorTable(hps, flOptions,
       lFormat, lStart, lCount, alTable);
```

GpiCreateLogColorTable Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCreateLogColorTable Parameter - flOptions

flOptions (ULONG) - input
Options.

This parameter can have one of the following values or 0:

LCOL_RESET
The color table is reset to its default values before processing the remainder of the data in this function.

This value is assumed if the color table is currently in RGB mode and is being changed to index mode; that is, LCOLF_INDRGB or LCOLF_CONSECRGB is specified.

The *lFormat* parameter must be LCOLF_INDRGB or LCOLF_CONSECRGB.

LCOL_PURECOLOR
When this option is set only colors for solid patterns (see GpiSetPattern) available in the physical palette will be used. Only pure colors are used and no dithering is done.

Other flags are reserved and must be 0.

GpiCreateLogColorTable Parameter - lFormat

lFormat (LONG) - input
Format of entries in the table.

This parameter can have one of the following values:

LCOLF_INDRGB
Array of index/RGB pairs. Each pair is 8 bytes long: 4 bytes (local format) for the index, and 4 bytes for the color value.

This sets the color table into index mode (and forces LCOL_RESET) if it is in RGB mode.

The maximum index that can be loaded is returned in the *CAPS_COLOR_INDEX* parameter of the DevQueryCaps function.

Each index specified must be greater than or equal to 0.

See "DevQueryCaps" in the *Presentation Manager Programming Reference* for more information.

LCOLF_CONSECRGB
Array of RGB values, corresponding to color indexes */Start* upwards. Each entry is 4 bytes long.

This sets the color table into index mode (and forces LCOL_RESET) if it is in RGB mode.

The maximum index that can be loaded is returned in the *CAPS_COLOR_INDEX* parameter of the DevQueryCaps function.

LCOLF_RGB
Color index = RGB.

This sets the color table into RGB mode.

GpiCreateLogColorTable Parameter - lStart

lStart (LONG) - input
Starting index.

This is relevant only for LCOLF_CONSECRGB.

The starting index must be greater than or equal to 0.

GpiCreateLogColorTable Parameter - lCount

lCount (LONG) - input
Count of elements in *alTable* .

This must be greater than or equal to 0. If 0 is specified, LCOLF_INDRGB and LCOLF_CONSECRGB have the same effect.

For LCOLF_INDRGB, *alTable* must contain an even number of elements. */Count* must be an even number.

GpiCreateLogColorTable Parameter - alTable

alTable (PLONG) - input
Start of the application data area.

This contains the color table definition data. The format depends on the value of *lFormat* .

Each color value is a 4-byte integer, with a value of

```
(R * 65536) + (G * 256) + B
```

where:

R	is red intensity value
G	is green intensity value
B	is blue intensity value. The maximum intensity for each primary is 255.

The high order byte must be 0.

GpiCreateLogColorTable Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiCreateLogColorTable - Parameters

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Options.

This parameter can have one of the following values or 0:

LCOL_RESET
The color table is reset to its default values before processing the remainder of the data in this function.

This value is assumed if the color table is currently in RGB mode and is being changed to index mode; that is, LCOLF_INDRGB or LCOLF_CONSECRGB is specified.

The *lFormat* parameter must be LCOLF_INDRGB or LCOLF_CONSECRGB.

LCOL_PURECOLOR
When this option is set only colors for solid patterns (see GpiSetPattern) available in the physical palette will be used. Only pure colors are used and no dithering is done.

Other flags are reserved and must be 0.

lFormat (LONG) - input
Format of entries in the table.

This parameter can have one of the following values:

LCOLF_INDRGB

Array of index/RGB pairs. Each pair is 8 bytes long: 4 bytes (local format) for the index, and 4 bytes for the color value.

This sets the color table into index mode (and forces LCOL_RESET) if it is in RGB mode.

The maximum index that can be loaded is returned in the *CAPS_COLOR_INDEX* parameter of the DevQueryCaps function.

Each index specified must be greater than or equal to 0.

See "DevQueryCaps" in the *Presentation Manager Programming Reference* for more information.

LCOLF_CONSECRGB
Array of RGB values, corresponding to color indexes *lStart* upwards. Each entry is 4 bytes long.

This sets the color table into index mode (and forces LCOL_RESET) if it is in RGB mode.

The maximum index that can be loaded is returned in the *CAPS_COLOR_INDEX* parameter of the DevQueryCaps function.

LCOLF_RGB
Color index = RGB.

This sets the color table into RGB mode.

lStart (LONG) - input
Starting index.

This is relevant only for LCOLF_CONSECRGB.

The starting index must be greater than or equal to 0.

lCount (LONG) - input
Count of elements in *alTable*.

This must be greater than or equal to 0. If 0 is specified, LCOLF_INDRGB and LCOLF_CONSECRGB have the same effect.

For LCOLF_INDRGB, *alTable* must contain an even number of elements. *lCount* must be an even number.

alTable (PLONG) - input
Start of the application data area.

This contains the color table definition data. The format depends on the value of *lFormat*.

Each color value is a 4-byte integer, with a value of

```
(R * 65536) + (G * 256) + B
```

where:

R	is red intensity value
G	is green intensity value
B	is blue intensity value. The maximum intensity for each primary is 255.

The high order byte must be 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiCreateLogColorTable - Remarks

This function can cause the color table to be reset to the default values. These are:

CLR_BACKGROUND	Reset color, used by GpiErase. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOWTEXT; see WinSetSysColors in the *Presentation Manager Programming Reference*). For a printer, it is the paper color. The background color for the display can be changed by setting new system colors from the Control Panel. The background color for a printer can be changed by selecting a new paper color (if allowed by the presentation driver).
CLR_BLUE	Blue.
CLR_RED	Red.
CLR_PINK	Pink (magenta).
CLR_GREEN	Green.
CLR_CYAN	Cyan (turquoise).
CLR_YELLOW	Yellow.
CLR_NEUTRAL	A device-dependent color that provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT; see WinSetSysColors). For a printer, it is a color that contrasts with the paper color. The neutral color for the display can be changed by setting new system colors from the Control Panel. The neutral color for a printer can be changed by selecting a new paper color (if allowed by the presentation driver).
CLR_DARKGRAY	Dark gray.
CLR_DARKBLUE	Dark blue.
CLR_DARKRED	Dark red.
CLR_DARKPINK	Dark pink.
CLR_DARKGREEN	Dark green.
CLR_DARKCYAN	Dark cyan.
CLR_BROWN	Brown.
CLR_PALEGRAY	Pale gray.

GpiErase clears the output of a device to the color defined by CLR_BACKGROUND.

By default, presentation spaces have a logical color table consisting of the 16 default values given above. In index mode, these entries are always considered as part of the color table, unless they are explicitly overwritten. Color indexes outside this range, which have not been loaded, are not considered as part of the color table; it is an error to use such colors if the color table is in index mode.

The system performs a mapping from the colors in the logical color table to those in the standard physical color table for that device. This mapping is used for all drawing and bit maps. Mixing is not predictable.

The standard physical color table always includes the standard 16 colors, where this is physically possible. On devices that support more than 16 colors, there may be additional colors available to which the requested colors may be mapped. However, it cannot be ensured that these additional colors are the same on different devices. Applications that depend upon precise colors beyond the first 16 should use a palette (see GpiCreatePalette) on devices for which this is supported. DevQueryCaps can be used to determine whether the function is supported by the device; see CAPS_PALETTE_MANAGER.

For a monochrome device (whether it is a display, bit map, printer, or some other type), a reset color is defined as follows:

1. Start with the appropriate item below:
 - The paper color, for a printer with no loaded color table
 - SYSCLR_WINDOW for a monochrome display with no loaded color table
 - Color 0, for any device if a color table has been loaded.
2. If this color is white or a light color, the reset color is set to white; otherwise, the reset color is set to black.

The reset color is used for:

- The color that GpiErase clears the output to
- CLR_BACKGROUND (color 0), unless an RGB color table is in use
- CLR_DEFAULT for GpiSetBackColor
- Any color that has exactly the same RGB value as the reset color.

Any other color becomes black if the reset color is white, and the converse.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiCreateLogColorTable - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COLOR_OPTIONS (0x2057)
An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COLOR_DATA (0x2054)
Invalid color table definition data was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_FORMAT (0x2055)
An invalid format parameter was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_START_INDEX (0x2058)
An invalid starting index parameter was specified with a logical color table or color query function.

PMERR_REALIZE_NOT_SUPPORTED (0x20F7)
An attempt was made to create a realizable logical color table on a device driver that does not support this function.

PMERR_PALETTE_SELECTED (0x210F)
Color palette operations cannot be performed on a presentation space while a palette is selected.

GpiCreateLogColorTable - Related Functions

Related Functions

- GpiCreatePalette
- GpiQueryColorData
- GpiQueryColorIndex
- GpiQueryLogColorTable
- GpiQueryNearestColor
- GpiQueryRealColors
- GpiQueryRGBColor

GpiCreateLogColorTable - Example Code

This example uses the GpiCreateLogColorTable function to create a logical color table, using data from the previous logical color table.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

HPS hps;                /* presentation space handle            */
LONG alTable[16];       /* assume 16 entries                    */

/* retrieve the current table */

GpiQueryLogColorTable(hps, 0L, 0L, 16L, alTable);

alTable[1] = 0x000080; /* change the second entry to light blue */

GpiCreateLogColorTable(hps,         /* presentation space       */
    0L,                             /* no special options       */
    LCOLF_CONSECRGB,                /* consecutive RGB values   */
    0L,                             /* start with color index 0 */
    16,                             /* 16 entries               */
    alTable);                       /* RGB color values         */
```

GpiCreateLogColorTable - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCreateLogFont

GpiCreateLogFont - Syntax

This function provides a logical definition of a font.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PSTR8      pName;      /*  Logical font name. */
LONG       lLcid;      /*  Local identifier. */
PFATTRS    pfatAttrs;  /*  Attributes required of the font. */
LONG       lMatch;     /*  Match indicators. */

lMatch = GpiCreateLogFont(hps, pName, lLcid,
          pfatAttrs);
```

GpiCreateLogFont Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiCreateLogFont Parameter - pName

pName (PSTR8) - input
Logical font name.

An 8-character name that can be used to describe the logical font. Its principal use is in interchange files, where it can help to identify the required font. If no interchange files are to be used, it can be null. For example, it can reference a file name that contains the font for a remote system.

GpiCreateLogFont Parameter - lLcid

lLcid (LONG) - input
Local identifier.

The local identifier that the application uses to refer to this font. It must be in the range 0 through 254. If 0 is specified, the properties of the default font are changed. The original default font can be restored by calling GpiDeleteSetId, with an *lLcid* parameter of LCID_DEFAULT or LCID_ALL.

If the *lLcid* parameter specifies a local identifier that is already being used to refer to a logical font, but is not the current pattern-set or marker-set local identifier, then the new definition replaces the old one. If *lLcid* specifies a local identifier that is already being used to refer to a logical font, and is the current pattern-set or marker-set local identifier, an error occurs. An error also occurs if the local identifier is currently used to refer to a bit map.

GpiCreateLogFont Parameter - pfatAttrs

pfatAttrs (PFATTRS) - input
Attributes required of the font.

GpiCreateLogFont Return Value - lMatch

lMatch (LONG) - returns
Match indicators.

FONT_MATCH
Font requirements matched successfully.
FONT_DEFAULT
Font requirements not matched; a default font is used.
GPI_ERROR
Error occurred.

GpiCreateLogFont - Parameters

hps (HPS) - input
Presentation-space handle.

pName (PSTR8) - input
Logical font name.

An 8-character name that can be used to describe the logical font. Its principal use is in interchange files, where it can help to identify the required font. If no interchange files are to be used, it can be null. For example, it can reference a file name that contains the font for a remote system.

lLcid (LONG) - input
Local identifier.

The local identifier that the application uses to refer to this font. It must be in the range 0 through 254. If 0 is specified, the properties of the default font are changed. The original default font can be restored by calling GpiDeleteSetId, with an *lLcid* parameter of LCID_DEFAULT or LCID_ALL.

If the *lLcid* parameter specifies a local identifier that is already being used to refer to a logical font, but is not the current pattern-set or marker-set local identifier, then the new definition replaces the old one. If *lLcid* specifies a local identifier that is already being used to refer to a logical font, and is the current pattern-set or marker-set local identifier, an error occurs. An error also occurs if the local identifier is currently used to refer to a bit map.

pfatAttrs (PFATTRS) - input
Attributes required of the font.

lMatch (LONG) - returns
Match indicators.

FONT_MATCH
Font requirements matched successfully.
FONT_DEFAULT
Font requirements not matched; a default font is used.
GPI_ERROR
Error occurred.

GpiCreateLogFont - Remarks

The system uses the available physical font that most closely matches the requirements. Physical fonts can be:

- Loaded at initialization time
- Built into particular devices or device drivers
- Private ones for this process, loaded by GpiLoadFonts.

An application can force selection of a particular physical font by quoting the *lMatch* value in FATTRS to be returned for the desired font by GpiQueryFonts. However, this method is only valid for a particular device/device driver combination on a single machine. This method should be avoided as a method for selecting fonts.

Whichever method is used, the choice of physical font, which is made when this function is issued, is never subsequently changed for a particular logical font.

The local identifier (*lLcid*) that the application decides to use to reference this logical font for later drawing operations is also specified; see GpiSetCharSet.

If the face name is provided, GpiCreateLogFont tries to select the font with that face name. If the face name is empty, GpiCreateLogFont selects a default font.

When a match number is provided, GpiCreateLogFont tries to find a font with the same match number and face name. If there is a mismatch at this point, GpiCreateLogFont acts as though the match number is 0 and starts the search again.

When the match number is 0 and the calling program requests a bit-map font (FATTR_FONTUSE_OUTLINE not set), GpiCreateLogFont searches for a bit-map font with the required average character width (AveCharWidth) and maximum baseline extent (MaxBaselineExt), consistent with the usage flags. If this search fails, GpiCreateLogFont searches for an outline font with the required face name.

When the match number is zero and the calling program requests an outline font (FATTR_FONTUSE_OUTLINE is set), GpiCreateLogFont searches for an outline font with the required selection flags. If that search fails, a default outline font is selected. If the match number is set to a positive number, a Presentation Manager font is selected. If the match number is negative, a font belonging to a physical device is selected.

It is advisable to set the values of *all* the elements in the *pfatAttrs* structure. This is particularly important where printing, plotting, or interchange are concerned, as the target machine may need to substitute an existing device font for the requested font.

To anticipate possible substitution by a vector font, values should be set for character angle, character shear and character box (using GpiSetCharAngle, GpiSetCharShear, and GpiSetCharBox respectively) before drawing any character strings. The GpiQueryFontMetrics function can be used to get the values of the character box height and width for a font. These are held in the fields *lEmHeight* and *lEmInc* in the FONTMETRICS structure.

Outline font characters are normally drawn filled; however, hollow characters are produced if the FATTR_SEL_OUTLINE flag is set in the *pfatAttrs* parameter. For small characters, outlining in this way can give a similar visual appearance to filled characters, with improved performance.

There are restrictions on the use of non-installed fonts with certain device types. See GpiLoadFonts for more details.

If this function occurs within a path definition when the drawing mode (see GpiSetDrawingMode) is **retain** or **draw-and-retain**, its effect is not stored with the definition.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiCreateLogFont - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SETID (0x20CA)
An invalid setid parameter was specified.

PMERR_INV_FONT_ATTRS (0x2072)
An invalid attrs parameter was specified with GpiCreateLogFont.

PMERR_FONT_NOT_LOADED (0x202F)
An attempt was made to create a font that was not loaded.

PMERR_SETID_IN_USE (0x2102)
An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set.

PMERR_KERNING_NOT_SUPPORTED (0x20D5)
Kerning was requested on GpiCreateLogFont call to a presentation space associated with a device context that does not support

kerning.

GpiCreateLogFont - Related Functions

Related Functions

- GpiDeleteSetId
- GpiLoadFonts
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryKerningPairs
- GpiQueryNumberSetIds
- GpiQuerySetIds
- GpiQueryWidthTable
- GpiSetCharMode
- GpiSetCharSet
- GpiSetMarkerSet
- GpiSetPatternSet
- GpiUnloadFonts

GpiCreateLogFont - Example Code

This example uses the GpiCreateLogFont function to create a logical font with the local identifier 1. The logical font has the face name "Courier" and requested width and height of 12 pels. Once the font is created, the example sets the font using the local identifier and displays a string in the font at the point (100,100).

```
#define INCL_GPILCIDS           /* Font functions               */
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptl = { 100, 100 };
FATTRS fat;

fat.usRecordLength = sizeof(FATTRS); /* sets size of structure   */
fat.fsSelection = 0;          /* uses default selection          */
fat.lMatch = 0L;              /* does not force match            */
fat.idRegistry = 0;           /* uses default registry           */
fat.usCodePage = 850;         /* code-page 850                   */
fat.lMaxBaselineExt = 12L;    /* requested font height is 12 pels */
fat.lAveCharWidth = 12L;      /* requested font width is 12 pels  */
fat.fsType = 0;               /* uses default type               */
fat.fsFontUse = FATTR_FONTUSE_NOMIX;/* doesn't mix with graphics */

/* Copy Courier to szFacename field */

strcpy(fat.szFacename ,"Courier");

GpiCreateLogFont(hps,         /* presentation space              */
                 NULL,        /* does not use logical font name  */
                 1L,          /* local identifier                */
                 &fat);       /* structure with font attributes  */

GpiSetCharSet(hps, 1L);       /* sets font for presentation space */
GpiCharStringAt(hps, &ptl, 5L, "Hello"); /* displays a string    */
```

GpiCreateLogFont - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCreatePalette

GpiCreatePalette - Syntax

This function creates and initializes a color palette.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB       hab;        /*  Anchor-block handle. */
ULONG     flOptions;  /*  Options. */
ULONG     ulFormat;   /*  Format of entries in the table. */
ULONG     ulCount;    /*  Count of elements in aulTable. */
PULONG    aulTable;   /*  Start of the application data area. */
HPAL      hpal;       /*  Palette handle. */

hpal = GpiCreatePalette(hab, flOptions, ulFormat,
         ulCount, aulTable);
```

GpiCreatePalette Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiCreatePalette Parameter - flOptions

flOptions (ULONG) - input
Options.

To combine the following two options, OR the values together. Other flags are reserved and must be 0.

LCOL_PURECOLOR

The application does not want color *dithering* to create colors not available in the physical palette for solid patterns (see GpiSetPattern). If this option is set, only pure colors are used and no dithering is done.

LCOL_OVERRIDE_DEFAULT_COLORS
Override option for applications that need the full hardware palette. The system does not guarantee a consistent look to the user interface when this option is used. The override is only in effect while the overriding palette is in the foreground.

GpiCreatePalette Parameter - ulFormat

ulFormat (ULONG) - input
Format of entries in the table.

It must be the following value:

LCOLF_CONSECRGB
Array of (RGB) values. Each entry is 4 bytes long. This is currently the only supported value for this parameter.

GpiCreatePalette Parameter - ulCount

ulCount (ULONG) - input
Count of elements in *aulTable* .

This must be greater than 0.

GpiCreatePalette Parameter - aulTable

aulTable (PULONG) - input
Start of the application data area.

This contains the palette definition data.

Each color value is a 4-byte integer, with a value of

```
(F * 16777216) + (R * 65536) + (G * 256) + B
```

where:

F is a flag byte, which can take the following values (these can be ORed together if required):

PC_RESERVED	This index is an animating index. This means that the application might frequently change the RGB value and so the system should not map the logical index of the palette of another application to the entry in the physical palette used for this color.
PC_EXPLICIT	The low-order word of the logical color table entry designates a physical palette slot from which the color definition is to be taken. This allows an application to show the actual contents of the device palette as realized for other logical palettes. This does not prevent the color in the slot from being changed for any reason.

R is red intensity value

G is green intensity value

B is blue intensity value. Each intensity value must be in the range 0 through 255.

GpiCreatePalette Return Value - hpal

hpal (HPAL) - returns
Palette handle.

<>0
Palette handle
GPI_ERROR
Error occurred.

GpiCreatePalette - Parameters

hab (HAB) - input
Anchor-block handle.

flOptions (ULONG) - input
Options.

To combine the following two options, OR the values together. Other flags are reserved and must be 0.

LCOL_PURECOLOR
The application does not want color *dithering* to create colors not available in the physical palette for solid patterns (see GpiSetPattern). If this option is set, only pure colors are used and no dithering is done.

LCOL_OVERRIDE_DEFAULT_COLORS
Override option for applications that need the full hardware palette. The system does not guarantee a consistent look to the user interface when this option is used. The override is only in effect while the overriding palette is in the foreground.

ulFormat (ULONG) - input
Format of entries in the table.

It must be the following value:

LCOLF_CONSECRGB
Array of (RGB) values. Each entry is 4 bytes long. This is currently the only supported value for this parameter.

ulCount (ULONG) - input
Count of elements in *aulTable* .

This must be greater than 0.

aulTable (PULONG) - input
Start of the application data area.

This contains the palette definition data.

Each color value is a 4-byte integer, with a value of

```
(F * 16777216) + (R * 65536) + (G * 256) + B
```

where:

F — is a flag byte, which can take the following values (these can be ORed together if required):

PC_RESERVED	This index is an animating index. This means that the application might frequently change the RGB value and so the system should not map the logical index of the palette of another application to the entry in the physical palette used for this color.
PC_EXPLICIT	The low-order word of the logical color table entry designates a physical palette slot from which the color definition is to be taken. This allows an application to show the actual contents of the device palette as realized for other logical palettes. This does not prevent the color in the slot from being changed for any reason.

R — is red intensity value

G — is green intensity value

B — is blue intensity value. Each intensity value must be in the range 0 through 255.

hpal (HPAL) - returns
Palette handle.

<>0
Palette handle

GPI_ERROR
Error occurred.

GpiCreatePalette - Remarks

The new palette contains only the entries set in the *aulTable* parameter. All color indices outside this range are not considered part of the palette; it is an error to use such colors when this palette is selected.

When a palette is realized (see WinRealizePalette in the *Presentation Manager Programming Reference*) the lowest indices are considered first. The palette should therefore be ordered so that the most important colors have the lowest indices. Animating indices, which on realization can have their own individual slots in the physical palette, should be used only when necessary.

Palettes should be created with only those color indices that the application requires and not unnecessarily create a large palette. The maximum index for a palette is not limited to CAPS_COLOR_INDEX.

The palette can be selected into a presentation space using GpiSelectPalette.

GpiCreatePalette - Errors

Possible returns from WinGetLastError

PMERR_INV_COLOR_OPTIONS (0x2057)
An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COLOR_DATA (0x2054)
Invalid color table definition data was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_FORMAT (0x2055)
An invalid format parameter was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_START_INDEX (0x2058)

An invalid starting index parameter was specified with a logical color table or color query function.

PMERR_INSUFFICIENT_MEMORY (0x203E)
The operation terminated through insufficient memory.

GpiCreatePalette - Related Functions

Related Functions

- GpiAnimatePalette
- GpiCreateLogColorTable
- GpiDeletePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSelectPalette
- GpiSetPaletteEntries

GpiCreatePalette - Example Code

The uses GpiCreatePalette to create and initialize a palette of 4 pure (no dithering) colors.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

HAB hab;                /* anchor block handle                  */
HPAL  hpal;             /* palette handle                       */
LONG  lFormat;          /* table entry format                   */

/************************************************************
 * assume 4 entries in palette.                              *
 * The RGB values are calculated with the following formula:     *
 *    (F * 16777216) + (R * 65536) + (G * 256) + B               *
 *    where F = flag, PC_RESERVED or PC_EXPLICIT                 *
 *          R = red intensity value                              *
 *          G = green intensity value                            *
 *          B = blue intensity value                             *
 * Thus, in the following table, red and green intensities are 0 *
 * while the blue intensity increases from 1 to 4.               *
 ************************************************************/

ULONG aulTable[4]=
          {(PC_RESERVED*16777216) + (0*65536) + (0*256) + 1,
           (PC_RESERVED*16777216) + (0*65536) + (0*256) + 2,
           (PC_RESERVED*16777216) + (0*65536) + (0*256) + 3,
           (PC_RESERVED*16777216) + (0*65536) + (0*256) + 4};


hpal = GpiCreatePalette(hab, 0L, LCOLF_CONSECRGB, 4L, aulTable);
```

GpiCreatePalette - Topics

Select an item:
Syntax
Parameters
Returns

Errors
Remarks
Example Code
Related Functions
Glossary

GpiCreatePS

GpiCreatePS - Syntax

This function creates a presentation space.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HAB       hab;        /*  Anchor-block handle. */
HDC       hdc;        /*  Device-context handle. */
PSIZEL    psizlSize;  /*  Presentation-page size. */
ULONG     flOptions;  /*  Options. */
HPS       hps;        /*  Presentation-space handle. */

hps = GpiCreatePS(hab, hdc, psizlSize, flOptions);
```

GpiCreatePS Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiCreatePS Parameter - hdc

hdc (HDC) - input
Device-context handle.

The handle of a device context with which the presentation space is to be associated, if GPIA_ASSOC is specified. This is mandatory for a micro presentation space (type GPIT_MICRO).

GpiCreatePS Parameter - psizlSize

psizlSize (PSIZEL) - input
Presentation-page size.

The size of the presentation page defines a rectangle in presentation page space, with the bottom-left corner at the origin. This rectangle is used for these purposes:

- Together with the page viewport, it defines the device transform. Whenever the presentation space is associated with a device context, a default page viewport is constructed, based on the presentation page size.
- It defines the "area of interest" of the picture. This is recorded in a metafile, if one is generated from this presentation space. Note, however, that depending upon the device transform, information drawn outside it may sometimes be visible; it is *not* a clipping boundary.
- If PU_ARBITRARY is specified, the page viewport is constructed such that the origin of the page rectangle maps to the origin of the default device rectangle (maximized window size, paper size, and so on), and either the right or top edges map, keeping the picture within the default device rectangle, and preserving its aspect ratio.

If 0 is specified as either the width or the height, GPIA_ASSOC must also be specified, and a presentation page of default dimension for the device (see above) is assumed. For PU_ARBITRARY the pel dimensions are used.

GpiCreatePS Parameter - flOptions

flOptions (ULONG) - input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

PS_UNITS

Presentation-page size units.

In each instance, the origin is at the bottom left.

One of these values *must* be specified:

PU_ARBITRARY	Application-convenient units
PU_PELS	Pel coordinates
PU_LOMETRIC	Units of 0.1 mm
PU_HIMETRIC	Units of 0.01 mm
PU_LOENGLISH	Units of 0.01 inch
PU_HIENGLISH	Units of 0.001 inch
PU_TWIPS	Units of 1/1440 inch.

PS_FORMAT

Coordinate format.

Indicates options to be used when storing coordinate values internally in the segment store.

For most calls, the format is not directly visible to an application. However, it is visible during editing (for example, GpiQueryElement). The format also has an effect on the amount of storage required for segment store. If a metafile is generated from this presentation space, the format also controls the format of the orders in the metafile.

Note: If GPIF_SHORT is selected, it is the responsibility of the application to ensure that the values passed for graphics coordinates are in the range -32 768 through +32 767, when the drawing mode is **retain** or **draw-and-retain** (see GpiSetDrawingMode), or if a metafile is being created. If in doubt, default or specify GPIF_LONG.

Do not specify GPIF_SHORT if a metafile of unknown format is to be played into this presentation space with GpiPlayMetaFile.

One of these can be selected, for a GPIT_NORMAL presentation space (for a GPIT_MICRO presentation space, only GPIF_DEFAULT is allowed):

GPIF_DEFAULT	Default local format (same as GPIF_LONG)
GPIF_SHORT	2-byte integers
GPIF_LONG	4-byte integers.

PS_TYPE

Presentation space.

Indicates the type of GPI presentation space required:

GPIT_NORMAL	Normal presentation space; this is the default
GPIT_MICRO	Micro presentation space.

Note: GPIA_ASSOC must also be set if GPIT_MICRO is set.

PS_MODE

Mode.

Reserved, must be 0 (default).

PS_ASSOCIATE

Association indicator.

Indicates whether the new presentation space is to be associated with the specified device context:

GPIA_NOASSOC	No association is required. This is the default.
GPIA_ASSOC	Association with *hdc* required.

Note: GPIA_ASSOC must be set if GPIT_MICRO is set.

PS_AREA

Area indicator.

Indicates whether the area definitions default to exclusive or inclusive; see *flOptions* in GpiBeginArea.

GPIM_AREAEXCL	Area definitions are exclusive.

GpiCreatePS Return Value - hps

hps (HPS) - returns
Presentation-space handle.

<>0
Presentation-space handle
GPI_ERROR
Error.

GpiCreatePS - Parameters

hab (HAB) - input
Anchor-block handle.

hdc (HDC) - input
Device-context handle.

The handle of a device context with which the presentation space is to be associated, if GPIA_ASSOC is specified. This is mandatory for a micro presentation space (type GPIT_MICRO).

psizlSize (PSIZEL) - input
Presentation-page size.

The size of the presentation page defines a rectangle in presentation page space, with the bottom-left corner at the origin. This rectangle is used for these purposes:

- Together with the page viewport, it defines the device transform. Whenever the presentation space is associated with a device context, a default page viewport is constructed, based on the presentation page size.
- It defines the "area of interest" of the picture. This is recorded in a metafile, if one is generated from this presentation space. Note, however, that depending upon the device transform, information drawn outside it may sometimes be visible; it is *not* a clipping boundary.
- If PU_ARBITRARY is specified, the page viewport is constructed such that the origin of the page rectangle maps to the origin of the default device rectangle (maximized window size, paper size, and so on), and either the right or top edges map, keeping the picture within the default device rectangle, and preserving its aspect ratio.

If 0 is specified as either the width or the height, GPIA_ASSOC must also be specified, and a presentation page of default dimension for the device (see above) is assumed. For PU_ARBITRARY the pel dimensions are used.

flOptions (ULONG) - input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

PS_UNITS

Presentation-page size units.

In each instance, the origin is at the bottom left.

One of these values *must* be specified:

PU_ARBITRARY	Application-convenient units
PU_PELS	Pel coordinates
PU_LOMETRIC	Units of 0.1 mm
PU_HIMETRIC	Units of 0.01 mm
PU_LOENGLISH	Units of 0.01 inch
PU_HIENGLISH	Units of 0.001 inch
PU_TWIPS	Units of 1/1440 inch.

PS_FORMAT

Coordinate format.

Indicates options to be used when storing coordinate values internally in the segment store.

For most calls, the format is not directly visible to an application. However, it is visible during editing (for example, GpiQueryElement). The format also has an effect on the amount of storage required for segment store. If a metafile is generated from this presentation space, the format also controls the format of the orders in the metafile.

Note: If GPIF_SHORT is selected, it is the responsibility of the application to ensure that the values passed for graphics coordinates are in the range -32 768 through +32 767, when the drawing mode is **retain** or **draw-and-retain** (see GpiSetDrawingMode), or if a metafile is being created. If in doubt, default or specify GPIF_LONG.

Do not specify GPIF_SHORT if a metafile of unknown format is to be played into this presentation space with GpiPlayMetaFile.

One of these can be selected, for a GPIT_NORMAL presentation space (for a GPIT_MICRO presentation space, only GPIF_DEFAULT is allowed):

GPIF_DEFAULT	Default local format (same as GPIF_LONG)
GPIF_SHORT	2-byte integers
GPIF_LONG	4-byte integers.

PS_TYPE

Presentation space.

Indicates the type of GPI presentation space required:

GPIT_NORMAL	Normal presentation space; this is the default

GPIT_MICRO — Micro presentation space.

Note: GPIA_ASSOC must also be set if GPIT_MICRO is set.

PS_MODE

Mode.

Reserved, must be 0 (default).

PS_ASSOCIATE

Association indicator.

Indicates whether the new presentation space is to be associated with the specified device context:

GPIA_NOASSOC — No association is required. This is the default.
GPIA_ASSOC — Association with *hdc* required.

Note: GPIA_ASSOC must be set if GPIT_MICRO is set.

PS_AREA

Area indicator.

Indicates whether the area definitions default to exclusive or inclusive; see *flOptions* in GpiBeginArea.

GPIM_AREAEXCL — Area definitions are exclusive.

hps (HPS) - returns
Presentation-space handle.

<>0
Presentation-space handle
GPI_ERROR
Error.

GpiCreatePS - Remarks

There are two types of presentation spaces:

- Micro presentation space
- Normal presentation space.

Only a restricted subset of calls is allowed to a micro presentation space; the main difference is that graphic segments (primitives, attributes, and so on) can be retained by the system, for subsequent redraw or editing, in a normal presentation space, but not in a micro presentation space. However, the storage and execution overheads are lower for a micro presentation space.

An initial association of the new presentation space with a device context may be performed (this is mandatory for a micro presentation space), by specifying GPIA_ASSOC.

When a presentation space is associated with a device context, either using this function with GPIA_ASSOC, or explicitly with GpiAssociate, a page viewport in device space is automatically constructed, to which the page is mapped to form the device transform. The value of *PS_UNITS* and the *psizlSize* parameter, are taken into account.

In general, the size parameter can be safely set to zeroes except when using PU_ARBITRARY units. In that case, use a size in device coordinates obtained from DevQueryCaps. For units other than PU_PELS, a non-zero size can cause a transform to be in effect for the resulting PS.

GpiCreatePS - Errors

Possible returns from WinGetLastError

PMERR_INV_OR_INCOMPAT_OPTIONS (0x20A9)
An invalid or incompatible (with micro presentation space) options parameter was specified with GpiCreatePS or GpiSetPS.

PMERR_DC_IS_ASSOCIATED (0x2017)
An attempt was made to associate a presentation space with a device context that was already associated or to destroy a device context that was associated.

PMERR_INV_HDC (0x207C)
An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

PMERR_INV_PS_SIZE (0x20BA)
An invalid size parameter was specified with GpiCreatePS or GpiSetPS.

GpiCreatePS - Related Functions

Related Functions

- GpiAssociate
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSavePS
- GpiSetPageViewport
- GpiSetPS

GpiCreatePS - Example Code

This example uses the GpiCreatePS function to create a micro presentation space for a memory device context. The function associates the presentation space with the device context and sets the page units to pels. By default, the presentation space is a normal presentation space that uses local storage format.

```
#define INCL_GPICONTROL         /* GPI control Functions        */
#include <os2.h>

HAB hab;                /* anchor block handle                  */
HDC hdc;                /* device context handle                */
HPS hps;                /* presentation space handle            */
SIZEL sizl = { 0, 0 };  /* use same page size as device         */
/*************************
 * context data structure *
 *************************/
DEVOPENSTRUC dop = {0L, "DISPLAY", NULL, 0L, 0L, 0L, 0L, 0L, 0L};

/* create memory device context */
hdc = DevOpenDC(hab, OD_MEMORY, "*", 5L, (PDEVOPENDATA)&dop, NULLHANDLE);

/* Create the presentation and associate the memory device
   context. */
hps = GpiCreatePS(hab, hdc, &sizl, PU_PELS |
                                   GPIT_MICRO | GPIA_ASSOC);
```

GpiCreatePS - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiCreateRegion

GpiCreateRegion - Syntax

This function creates a region, for a particular class of device, using a series of rectangles.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;             /*  Presentation-space handle. */
LONG      lCount;          /*  The number of rectangles. */
PRECTL    arclRectangles;  /*  An array of rectangles. */
HRGN      hrgn;            /*  Region handle. */

hrgn = GpiCreateRegion(hps, lCount, arclRectangles);
```

GpiCreateRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

A region suitable for use with the currently associated device is created.

GpiCreateRegion Parameter - lCount

lCount (LONG) - input
The number of rectangles.

The number specified in *arclRectangles* . If *lCount* is 0, an empty region is created, and *arclRectangles* is ignored.

GpiCreateRegion Parameter - arclRectangles

arclRectangles (PRECTL) - input
An array of rectangles.

The rectangles are specified in device coordinates.

For each rectangle in the array, the value of *xRight* must be greater than (or equal to) *xLeft*, and *yTop* must be greater than (or equal to) *yBottom*.

GpiCreateRegion Return Value - hrgn

hrgn (HRGN) - returns
Region handle.

<>0
Region handle
RGN_ERROR
Error.

GpiCreateRegion - Parameters

hps (HPS) - input
Presentation-space handle.

A region suitable for use with the currently associated device is created.

lCount (LONG) - input
The number of rectangles.

The number specified in *arclRectangles*. If *lCount* is 0, an empty region is created, and *arclRectangles* is ignored.

arclRectangles (PRECTL) - input
An array of rectangles.

The rectangles are specified in device coordinates.

For each rectangle in the array, the value of *xRight* must be greater than (or equal to) *xLeft*, and *yTop* must be greater than (or equal to) *yBottom*.

hrgn (HRGN) - returns
Region handle.

<>0
Region handle
RGN_ERROR
Error.

GpiCreateRegion - Remarks

The new region is defined by the logical-OR of all of the rectangles specified. Points on the right-hand and top boundaries are not included in the region. Points on the left-hand and bottom boundaries, that are not also on the right-hand or top boundaries (that is, the top-left and bottom-right corner points), are included.

The region is owned by the process from which this function is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system.

GpiCreateRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
: An invalid rectangle parameter was specified.

GpiCreateRegion - Related Functions

Related Functions

- GpiCombineRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

GpiCreateRegion - Example Code

This example uses the GpiCreateRegion function to create a region consisting of the union of three rectangles.

```
#define INCL_GPIREGIONS         /* Region functions            */
#include <os2.h>

HPS hps;                /* presentation space handle           */
HRGN hrgn;                              /* handle for region */
RECTL arcl[3] = { 100, 100, 200, 200,      /* 1st rectangle     */
                  150, 150, 250, 250,      /* 2nd rectangle     */
                  200, 200, 300, 300 };    /* 3rd rectangle     */
```

```
hrgn = GpiCreateRegion(hps,              /* presentation space */
                       3L,               /* three rectangles   */
                       arcl); /* address of array of rectangles */
```

GpiCreateRegion - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeleteBitmap

GpiDeleteBitmap - Syntax

This function deletes a bit map.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HBITMAP    hbm;  /*  Handle of the bit map to be deleted. */
BOOL       rc;   /*  Success indicator. */

rc = GpiDeleteBitmap(hbm);
```

GpiDeleteBitmap Parameter - hbm

hbm (HBITMAP) - input
Handle of the bit map to be deleted.

GpiDeleteBitmap Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteBitmap - Parameters

hbm (HBITMAP) - input
Handle of the bit map to be deleted.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteBitmap - Remarks

There are restrictions on the use of this function while generating a metafile or a PM_Q_STD print file; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiDeleteBitmap - Errors

Possible returns from WinGetLastError

PMERR_INV_HBITMAP (0x207B)
An invalid bit-map handle was specified.

PMERR_BITMAP_IS_SELECTED (0x2009)
An attempt was made to delete a bit map while it was selected into a device context.

PMERR_HBITMAP_BUSY (0x2032)
An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiDeleteBitmap - Related Functions

Related Functions

- GpiBitBlt

- GpiCreateBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt

GpiDeleteBitmap - Example Code

This example uses the GpiDeleteBitmap function to delete a bit map. The GpiSetBitmap function releases the bit map from the presentation space before deleting it. This is needed only if the bit map is set in the presentation space.

```
#define INCL_GPIBITMAPS         /* GPI Bit map functions        */
#include <os2.h>

HPS hps;                /* presentation space handle            */
HBITMAP hbm, hbmPrevious;

hbm = GpiLoadBitmap(hps, 0L, 1, 0L, 0L);  /* load the bit map   */
hbmPrevious = GpiSetBitmap(hps, hbm);     /* set bit map for PS */

/* bit map displayed with GpiBitBlt */

GpiSetBitmap(hps, hbmPrevious);      /* release bit map from PS */
GpiDeleteBitmap(hbm);                /* delete the bit map      */
```

GpiDeleteBitmap - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeleteElement

GpiDeleteElement - Syntax

This function deletes the element indicated by the element pointer, that was set with GpiSetElementPointer or GpiSetElementPointerAtLabel.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiDeleteElement(hps);
```

GpiDeleteElement Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDeleteElement Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteElement - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteElement - Remarks

The element pointer is set to the element immediately preceding the deleted element.

If the element pointer has a value of 0 (points that are logically before the first element), nothing is deleted and the element pointer is not changed.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is invalid within an element bracket.

GpiDeleteElement - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_NO_CURRENT_SEG (0x20E6)
An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

GpiDeleteElement - Related Functions

Related Functions

- GpiBeginElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

GpiDeleteElement - Example Code

This example uses the GpiDeleteElement function to delete the third element from the previously created segment 2.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#include <os2.h>

HPS  hps;
```

```
GpiOpenSegment(hps, 2L);           /* open segment #2       */
GpiSetElementPointer(hps, 3L);     /* move to third element */
GpiDeleteElement(hps);             /* delete element        */
GpiCloseSegment(hps);              /* close the segment     */
```

GpiDeleteElement - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeleteElementRange

GpiDeleteElementRange - Syntax

This function deletes all elements between, and including, the elements indicated by the specified element numbers.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;            /*  Presentation-space handle. */
LONG    lFirstElement;  /*  Number of the first element to be deleted. */
LONG    lLastElement;   /*  Number of the last element to be deleted. */
BOOL    rc;             /*  Success indicator. */

rc = GpiDeleteElementRange(hps, lFirstElement,
       lLastElement);
```

GpiDeleteElementRange Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDeleteElementRange Parameter - lFirstElement

lFirstElement (LONG) - input
Number of the first element to be deleted.

If lFirstElement > lLastElement, the values are swapped before being used.

GpiDeleteElementRange Parameter - lLastElement

lLastElement (LONG) - input
Number of the last element to be deleted.

If lLastElement < lFirstElement, the values are swapped before being used.

GpiDeleteElementRange Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteElementRange - Parameters

hps (HPS) - input
Presentation-space handle.

lFirstElement (LONG) - input
Number of the first element to be deleted.

If lFirstElement > lLastElement, the values are swapped before being used.

lLastElement (LONG) - input
Number of the last element to be deleted.

If lLastElement < lFirstElement, the values are swapped before being used.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteElementRange - Remarks

If either element number is outside the range of the current segment, it is set to the nearest valid value.

When this function has finished, the element pointer is set to the element immediately preceding the deleted element or elements.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is not valid within an element bracket.

GpiDeleteElementRange - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
: An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_NO_CURRENT_SEG (0x20E6)
: An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

PMERR_INV_IN_ELEMENT (0x2089)
: An attempt was made to issue a function invalid inside an element bracket.

GpiDeleteElementRange - Related Functions

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

GpiDeleteElementRange - Example Code

This example uses the GpiDeleteElementRange function to delete the second through fifth elements in the previously created segment 2.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#include <os2.h>

HPS  hps;

GpiOpenSegment(hps, 2L);           /* open segment # 2           */
GpiDeleteElementRange(hps, 2L, 5L);/* delete elements 2 thru 5   */
GpiCloseSegment(hps);              /* close the segment          */
```

GpiDeleteElementRange - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeleteElementsBetweenLabels

GpiDeleteElementsBetweenLabels - Syntax

This function deletes all elements between, but not including, the elements found to contain the indicated labels.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;          /*  Presentation-space handle. */
LONG    lFirstLabel;  /*  Label marking the start of the elements to be deleted. */
LONG    lLastLabel;   /*  Label marking the end of the elements to be deleted. */
BOOL    rc;           /*  Success indicator. */

rc = GpiDeleteElementsBetweenLabels(hps, lFirstLabel,
       lLastLabel);
```

GpiDeleteElementsBetweenLabels Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDeleteElementsBetweenLabels Parameter - lFirstLabel

lFirstLabel (LONG) - input
Label marking the start of the elements to be deleted.

It must be greater or equal to 0.

GpiDeleteElementsBetweenLabels Parameter - lLastLabel

lLastLabel (LONG) - input
Label marking the end of the elements to be deleted.

It must be greater or equal to 0.

GpiDeleteElementsBetweenLabels Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteElementsBetweenLabels - Parameters

hps (HPS) - input
Presentation-space handle.

lFirstLabel (LONG) - input
Label marking the start of the elements to be deleted.

It must be greater or equal to 0.

lLastLabel (LONG) - input
Label marking the end of the elements to be deleted.

It must be greater or equal to 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE

Error occurred.

GpiDeleteElementsBetweenLabels - Remarks

The search for *lFirstLabel* and *lLastLabel* is performed separately, and starts from the element pointed to by the current element pointer.

See also:

- GpiSetElementPointer
- GpiSetElementPointerAtLabel.

If either label cannot be found between the current element pointer location and the end of the segment, an error is generated and no deletion occurs.

On completion, the element pointer is set to the element immediately preceding the deleted elements.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is not valid within an element bracket.

GpiDeleteElementsBetweenLabels - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_NO_CURRENT_SEG (0x20E6)
An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

PMERR_LABEL_NOT_FOUND (0x20D6)
The specified element label did not exist.

GpiDeleteElementsBetweenLabels - Related Functions

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer

- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

GpiDeleteElementsBetweenLabels - Example Code

This example uses the GpiDeleteElementsBetweenLabels function to delete the elements between, but not including, the elements having the labels 1 and 2.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#include <os2.h>

HPS  hps;

GpiOpenSegment(hps, 2L);            /* open segment #2   */

/* delete elements between 1 and 2 */

GpiDeleteElementsBetweenLabels(hps, 1L, 2L);
GpiCloseSegment(hps);               /* close the segment */
```

GpiDeleteElementsBetweenLabels - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeleteMetaFile

GpiDeleteMetaFile - Syntax

This function deletes a metafile.

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HMF     hmf;  /*  Metafile handle. */
BOOL    rc;   /*  Success indicator. */
```

```
rc = GpiDeleteMetaFile(hmf);
```

GpiDeleteMetaFile Parameter - hmf

hmf (HMF) - input
Metafile handle.

GpiDeleteMetaFile Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteMetaFile - Parameters

hmf (HMF) - input
Metafile handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteMetaFile - Remarks

This function deletes access to the specified memory metafile and makes the metafile handle invalid.

GpiDeleteMetaFile - Errors

Possible returns from WinGetLastError

PMERR_INV_HMF (0x207E)
An invalid metafile handle was specified.

PMERR_METAFILE_IN_USE (0x20D9)
An attempt has been made to access a metafile that is in use by another thread.

PMERR_TOO_MANY_METAFILES_IN_USE (0x2106)
The maximum number of metafiles allowed for a given process was exceeded.

GpiDeleteMetaFile - Related Functions

Related Functions

- GpiCopyMetaFile
- GpiLoadMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile
- GpiSetMetaFileBits

GpiDeleteMetaFile - Example Code

This example uses GpiDeleteMetaFile to delete a metafile previously loaded with GpiLoadMetaFile.

```
#define INCL_GPIMETAFILES       /* Metafile functions           */
#include <os2.h>

BOOL     fSuccess;      /* success indicator                    */
HMF      hmf;           /* metafile handle                      */
HAB hab;                /* anchor block handle                  */

/* loads metafile from disk */
hmf = GpiLoadMetaFile(hab, "sample.met");
    .
    .
    .

fSuccess = GpiDeleteMetaFile(hmf);
```

GpiDeleteMetaFile - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeletePalette

GpiDeletePalette - Syntax

This function deletes a color palette.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPAL    hpal;  /*  Palette handle. */
BOOL    rc;    /*  Success indicator. */

rc = GpiDeletePalette(hpal);
```

GpiDeletePalette Parameter - hpal

hpal (HPAL) - input
Palette handle.

GpiDeletePalette Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeletePalette - Parameters

hpal (HPAL) - input
Palette handle.

rc (BOOL) - returns
Success indicator.

TRUE

TRUE
: Successful completion

FALSE
: Error occurred.

GpiDeletePalette - Remarks

The palette must not be currently selected into a presentation space (see GpiSelectPalette).

GpiDeletePalette - Errors

Possible returns from WinGetLastError

PMERR_INV_HPAL (0x2111)
: An invalid color palette handle was specified.

PMERR_PALETTE_SELECTED (0x210F)
: Color palette operations cannot be performed on a presentation space while a palette is selected.

PMERR_PALETTE_BUSY (0x2112)
: An attempt has been made to reset the owner of a palette when it was busy.

GpiDeletePalette - Related Functions

Related Functions

- GpiAnimatePalette
- GpiCreatePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSelectPalette
- GpiSetPaletteEntries

GpiDeletePalette - Example Code

This example uses GpiDeletePalette to delete the color palette currently associated with the presentation space, which is determined using GpiQueryPalette.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

BOOL     fSuccess;      /* success indicator                   */
HPAL     hpal;          /* palette handle                      */
HPS      hps;           /* Presentation-space handle           */

/* get handle of currently associated palette */
hpal = GpiQueryPalette(hps);

/* delete palette */
fSuccess = GpiDeletePalette(hpal);
```

GpiDeletePalette - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeleteSegment

GpiDeleteSegment - Syntax

This function deletes a retained segment.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lSegid;  /*  Segment identifier. */
BOOL    rc;      /*  Success indicator. */

rc = GpiDeleteSegment(hps, lSegid);
```

GpiDeleteSegment Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDeleteSegment Parameter - lSegid

lSegid (LONG) - input
Segment identifier.

The identifier of the segment to be deleted; it must be greater than 0.

GpiDeleteSegment Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteSegment - Parameters

hps (HPS) - input
Presentation-space handle.

lSegid (LONG) - input
Segment identifier.

The identifier of the segment to be deleted; it must be greater than 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteSegment - Remarks

If the segment is open when it is deleted, there is no open segment after this function. In this instance, processing as described for GpiCloseSegment is performed.

If the segment is in the segment chain, it is removed from the chain.

This function deletes only a retained segment.

Note: In **draw** drawing mode (see GpiSetDrawingMode), the identifier of the current segment is not remembered, so it is not recognized if specified as the *lSegid* parameter.

GpiDeleteSegment - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiDeleteSegment - Related Functions

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

GpiDeleteSegment - Example Code

This example uses the GpiDeleteSegment function to delete segment 4, previously created by GpiOpenSegment.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiOpenSegment(hps, 4L);              /* open the segment           */
GpiMove(hps, &ptlStart);              /* move to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle);    /* draw triangle              */
GpiCloseSegment(hps);                 /* close the segment          */
    .
    .
    .
GpiDeleteSegment(hps, 4L);            /* delete segment #4          */
```

GpiDeleteSegment - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeleteSegments

GpiDeleteSegments - Syntax

This function deletes all segments in the given identifier range.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;            /*  Presentation-space handle. */
LONG    lFirstSegment;  /*  First identifier in the range; it must be greater than 0. */
LONG    lLastSegment;   /*  Last identifier in the range; it must be greater than 0. */
BOOL    rc;             /*  Success indicator. */

rc = GpiDeleteSegments(hps, lFirstSegment,
       lLastSegment);
```

GpiDeleteSegments Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDeleteSegments Parameter - lFirstSegment

lFirstSegment (LONG) - input
First identifier in the range; it must be greater than 0.

GpiDeleteSegments Parameter - lLastSegment

lLastSegment (LONG) - input
Last identifier in the range; it must be greater than 0.

GpiDeleteSegments Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteSegments - Parameters

hps (HPS) - input
Presentation-space handle.

lFirstSegment (LONG) - input
First identifier in the range; it must be greater than 0.

lLastSegment (LONG) - input
Last identifier in the range; it must be greater than 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteSegments - Remarks

lFirstSegment and *lLastSegment* can have the same value, in which instance, only this segment is deleted. If *lFirstSegment* is greater than *lLastSegment* only the segment with identifier *lFirstSegment* is deleted.

If one of the segments deleted is the currently open segment, there is no open segment after this function. In this instance, processing as described for GpiCloseSegment is performed. If any of the segments are in the segment chain, they are removed from the chain.

This function only deletes retained segments.

Note: In **draw** drawing mode (see GpiSetDrawingMode), the identifier of the current segment is not remembered, so it is not recognized if it occurs within the range of specified identifiers.

GpiDeleteSegments - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiDeleteSegments - Related Functions

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

GpiDeleteSegments - Example Code

This example uses the GpiDeleteSegments function to delete segments 4 through 6, created by GpiOpenSegment.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS hps;                /* presentation space handle            */

GpiOpenSegment(hps, 4L);              /* open segment 4         */
    .
GpiCloseSegment(hps);                 /* close the segment      */
GpiOpenSegment(hps, 5L);              /* open segment 5         */
    .
GpiCloseSegment(hps);                 /* close the segment      */
GpiOpenSegment(hps, 6L);              /* open segment 6         */
    .
GpiCloseSegment(hps);                 /* close the segment      */
    .
    .
    .
GpiDeleteSegments(hps, 4L, 6L);  /* delete segments 4 through 6 */
```

GpiDeleteSegments - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDeleteSetId

GpiDeleteSetId - Syntax

This function deletes a logical font or bit-map tag.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lLcid;  /*  Local identifier. */
BOOL    rc;     /*  Success indicator. */

rc = GpiDeleteSetId(hps, lLcid);
```

GpiDeleteSetId Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDeleteSetId Parameter - lLcid

lLcid (LONG) - input
Local identifier.

The local identifier (lcid) for the object. It must be in the range of 0 through 254.

If LCID_ALL is specified, **all** logical fonts are deleted, and all bit-map tagging is removed. If LCID_DEFAULT or LCID_ALL is specified, the original default font is restored if it has been changed (see GpiCreateLogFont).

GpiDeleteSetId Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteSetId - Parameters

hps (HPS) - input
Presentation-space handle.

lLcid (LONG) - input
Local identifier.

The local identifier (lcid) for the object. It must be in the range of 0 through 254.

If LCID_ALL is specified, **all** logical fonts are deleted, and all bit-map tagging is removed. If LCID_DEFAULT or LCID_ALL is specified, the original default font is restored if it has been changed (see GpiCreateLogFont).

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDeleteSetId - Remarks

If the object is a logical font, it is deleted, and is no longer available for use. If the object is a bit map, it is no longer tagged with the local identifier; the bit map is not deleted and its handle remains valid.

In either instance, the *lLcid* is released and is now available for reuse, unless the object is currently selected (as the current character, pattern, or marker set), in which instance an error is raised.

If this function occurs within a path definition when the drawing mode (see GpiSetDrawingMode) is **retain** or **draw-and-retain**, its effect is not stored with the definition.

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiDeleteSetId - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SETID (0x20CA)
: An invalid setid parameter was specified.

PMERR_SETID_NOT_FOUND (0x2103)
: An attempt was made to delete a setid that did not exist.

PMERR_SETID_IN_USE (0x2102)
: An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set.

GpiDeleteSetId - Related Functions

Related Functions

- GpiCreateLogFont
- GpiLoadFonts
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryKerningPairs
- GpiQueryNumberSetIds
- GpiQuerySetIds
- GpiQueryWidthTable
- GpiSetBitmapId
- GpiSetCharSet
- GpiUnloadFonts

GpiDeleteSetId - Example Code

This example uses the GpiDeleteSetId function to delete a logical font. The GpiSetCharSet function is required only if the logical font is the current font for the presentation space.

```
#define INCL_GPILCIDS           /* Font functions               */
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
FATTRS fat;

/* create and set the font */

GpiCreateLogFont(hps, NULL, 1L, &fat);
GpiSetCharSet(hps, 1L);
    .
    .
    .
GpiSetCharSet(hps, 0L);          /* release the font before deleting */
GpiDeleteSetId(hps, 1L);         /* delete the logical font          */
```

GpiDeleteSetId - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDestroyPS

GpiDestroyPS - Syntax

This function destroys the presentation space.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiDestroyPS(hps);
```

GpiDestroyPS Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDestroyPS Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiDestroyPS - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDestroyPS - Remarks

All resources owned by the presentation space are released, and any subsequent calls that use the value of the presentation space handle are rejected.

GpiDestroyPS - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_PS_IS_ASSOCIATED (0x20F5)
An attempt was made to destroy a presentation or associate a presentation space that is still associated with a device context.

GpiDestroyPS - Related Functions

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSavePS
- GpiSetPS

GpiDestroyPS - Example Code

This example uses the GpiDestroyPS function to destroy the presentation space associated with a memory device context.

```
#define INCL_GPICONTROL         /* GPI control Functions        */
#define INCL_DEV                /* Device Function definitions  */
#include <os2.h>

HAB          hab;       /* Anchor-block handle                  */
HPS          hps;       /* Target presentation-space handle     */
HDC          hdc;       /* Device-context handle                */
DEVOPENSTRUC dop;       /* context data structure               */
SIZEL page = { 0, 0 };  /* page size (use same as device)       */

/* Create the memory device context and presentation space. */
hdc = DevOpenDC(hab, OD_MEMORY, "*", 5L, (PDEVOPENDATA)&dop, NULLHANDLE);
hps = GpiCreatePS(hab, hdc, &page, PU_PELS|GPIT_MICRO|GPIA_ASSOC);
    .
    .
    .
GpiAssociate(hps, NULLHANDLE); /* disassociate device context */
GpiDestroyPS(hps);       /* destroys presentation space */
DevCloseDC(hdc);         /* closes device context       */
```

GpiDestroyPS - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDestroyRegion

GpiDestroyRegion - Syntax

This function destroys a region.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
HRGN    hrgn;  /*  Handle of region to be destroyed. */
```

```
BOOL    rc;    /*  Success indicator. */
rc = GpiDestroyRegion(hps, hrgn);
```

GpiDestroyRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

GpiDestroyRegion Parameter - hrgn

hrgn (HRGN) - input
Handle of region to be destroyed.

If this is NULLHANDLE, the call takes no action, and completes without error.

GpiDestroyRegion Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDestroyRegion - Parameters

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) - input
Handle of region to be destroyed.

If this is NULLHANDLE, the call takes no action, and completes without error.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiDestroyRegion - Remarks

This function cannot be used to destroy the clip region; the clip region must first be deselected with GpiSetClipRegion.

GpiDestroyRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
: An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
: An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY (0x2034)
: An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiDestroyRegion - Related Functions

Prerequisite Functions

- GpiSetClipRegion(if the region to be destroyed is a clip region)

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

GpiDestroyRegion - Example Code

This example uses the GpiDestroyRegion function to destroy a region after drawing a complex figure.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>

HPS hps;                /* presentation space handle            */
HRGN hrgn;
RECTL arcl[3] = { 10,10,20,20,15,15,25,25,20,20,30,30 };

hrgn = GpiCreateRegion(hps, 3L, arcl); /* use 3 rectangles   */
GpiPaintRegion(hps, hrgn);             /* paint the region   */
GpiDestroyRegion(hps, hrgn);           /* destroy the region */
```

GpiDestroyRegion - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDrawBits

GpiDrawBits - Syntax

This function draws a rectangle of bits.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS             hps;            /*  Target presentation-space handle. */
PVOID           pBits;          /*  Source bits. */
PBITMAPINFO2    pbmiInfoTable;  /*  Bit-map information table. */
LONG            lCount;         /*  Point count. */
PPOINTL         aptlPoints;     /*  Point array. */
LONG            lRop;           /*  Mixing function required. */
ULONG           flOptions;      /*  Options. */
LONG            lHits;          /*  Correlation and error indicators. */

lHits = GpiDrawBits(hps, pBits, pbmiInfoTable,
          lCount, aptlPoints, lRop, flOptions);
```

GpiDrawBits Parameter - hps

hps (HPS) - input
Target presentation-space handle.

GpiDrawBits Parameter - pBits

pBits (PVOID) - input
Source bits.

The source bits must be in one of the standard bit-map formats. This field must point to a doubleword aligned address to assure correct creation of the bit map by the device in device specific format.

GpiDrawBits Parameter - pbmiInfoTable

pbmiInfoTable (PBITMAPINFO2) - input
Bit-map information table.

This describes the format of the source bits.

GpiDrawBits Parameter - lCount

lCount (LONG) - input
Point count.

This count must be equal to 4.

GpiDrawBits Parameter - aptlPoints

aptlPoints (PPOINTL) - input
Point array.

Array of *lCount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**. These are:

Tx1,Ty1
Specify the bottom left corner of the target rectangle in target world coordinates.

Tx2,Ty2
Specify the top right corner of the target rectangle in target world coordinates.

An error occurs if the left coordinate of the target rectangle is greater than the right, or if the bottom coordinate is greater than the top.

Sx1,Sy1
: Specify the bottom left corner of the source rectangle in source device coordinates.

Sx2,Sy2
: Specify the top right corner of the source rectangle in source device coordinates.

GpiDrawBits Parameter - lRop

lRop (LONG) - input
Mixing function required.

Each plane of the target can be considered to be processed separately. For any pel in a target plane, three bits together with the *lRop* values are used to determine the final value. These are the value of that pel in the Pattern (P) and Source (S) data and the initial value of that pel in the Target (T) data. For any combination of P, S, and T pel values, the final target value for the pel is determined by the appropriate *lRop* bit value as shown below:

P	S	T(initial)	T(final)
0	0	0	Index bit 0 (least significant)
0	0	1	Index bit 1
0	1	0	Index bit 2
0	1	1	Index bit 3
1	0	0	Index bit 4
1	0	1	Index bit 5
1	1	0	Index bit 6
1	1	1	Index bit 7 (most significant)

The index formed as described above determines the mixing required. Mnemonic names are available for commonly used mixes:

```
ROP_SRCCOPY       /* SRC                     */
ROP_SRCPAINT      /* SRC OR DST              */
ROP_SRCAND        /* SRC AND DST             */
ROP_SRCINVERT     /* SRC XOR DST             */
ROP_SRCERASE      /* SRC AND NOT(DST)        */
ROP_NOTSRCCOPY    /* NOT(SRC)                */
ROP_NOTSRCERASE   /* NOT(SRC) AND NOT(DST)   */
ROP_MERGECOPY     /* SRC AND PAT             */
ROP_MERGEPAINT    /* NOT(SRC) OR DST         */
ROP_PATCOPY       /* PAT                     */
ROP_PATPAINT      /* NOT(SRC) OR PAT OR DST  */
ROP_PATINVERT     /* DST XOR PAT             */
ROP_DSTINVERT     /* NOT(DST)                */
ROP_ZERO          /* 0                       */
ROP_ONE           /* 1                       */
```

GpiDrawBits Parameter - flOptions

flOptions (ULONG) - input

Options.

How eliminated lines or columns are treated if a compression is performed.

Flags 15 through 31 of *flOptions* can be used for privately supported modes for particular devices.

BBO_OR
: The default value; if compression is necessary, logical-OR eliminated rows or columns. This is useful for white on black.

BBO_AND
: If compression is necessary, logical-AND eliminated rows or columns. This is useful for black on white.

BBO_IGNORE
: If compression is necessary, ignore eliminated rows or columns. This is useful for color.

BBO_PAL_COLORS
: The color table passed in with the Bitmap-InfoTable is defined as indices into the currently realized palette, rather than actual colors.

GpiDrawBits Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiDrawBits - Parameters

hps (HPS) - input
Target presentation-space handle.

pBits (PVOID) - input
Source bits.

The source bits must be in one of the standard bit-map formats. This field must point to a doubleword aligned address to assure correct creation of the bit map by the device in device specific format.

pbmiInfoTable (PBITMAPINFO2) - input
Bit-map information table.

This describes the format of the source bits.

lCount (LONG) - input
Point count.

This count must be equal to 4.

aptlPoints (PPOINTL) - input
Point array.

Array of *lCount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**. These are:

Tx1,Ty1
: Specify the bottom left corner of the target rectangle in target world coordinates.

Tx2,Ty2
: Specify the top right corner of the target rectangle in target world coordinates.

 An error occurs if the left coordinate of the target rectangle is greater than the right, or if the bottom coordinate is greater than the top.

Sx1,Sy1
: Specify the bottom left corner of the source rectangle in source device coordinates.

Sx2,Sy2
: Specify the top right corner of the source rectangle in source device coordinates.

lRop (LONG) - input
Mixing function required.

Each plane of the target can be considered to be processed separately. For any pel in a target plane, three bits together with the *lRop* values are used to determine the final value. These are the value of that pel in the Pattern (P) and Source (S) data and the initial value of that pel in the Target (T) data. For any combination of P, S, and T pel values, the final target value for the pel is determined by the appropriate *lRop* bit value as shown below:

P	S	T(initial)	T(final)
0	0	0	Index bit 0 (least significant)
0	0	1	Index bit 1
0	1	0	Index bit 2
0	1	1	Index bit 3
1	0	0	Index bit 4
1	0	1	Index bit 5
1	1	0	Index bit 6
1	1	1	Index bit 7 (most significant)

The index formed as described above determines the mixing required. Mnemonic names are available for commonly used mixes:

```
ROP_SRCCOPY       /* SRC                     */
ROP_SRCPAINT      /* SRC OR DST              */
ROP_SRCAND        /* SRC AND DST             */
ROP_SRCINVERT     /* SRC XOR DST             */
ROP_SRCERASE      /* SRC AND NOT(DST)        */
ROP_NOTSRCCOPY    /* NOT(SRC)                */
ROP_NOTSRCERASE   /* NOT(SRC) AND NOT(DST)   */
ROP_MERGECOPY     /* SRC AND PAT             */
ROP_MERGEPAINT    /* NOT(SRC) OR DST         */
ROP_PATCOPY       /* PAT                     */
ROP_PATPAINT      /* NOT(SRC) OR PAT OR DST  */
ROP_PATINVERT     /* DST XOR PAT             */
ROP_DSTINVERT     /* NOT(DST)                */
ROP_ZERO          /* 0                       */
ROP_ONE           /* 1                       */
```

flOptions (ULONG) - input
Options.

How eliminated lines or columns are treated if a compression is performed.

Flags 15 through 31 of *flOptions* can be used for privately supported modes for particular devices.

BBO_OR
: The default value; if compression is necessary, logical-OR eliminated rows or columns. This is useful for white on black.

BBO_AND
: If compression is necessary, logical-AND eliminated rows or columns. This is useful for black on white.

BBO_IGNORE
: If compression is necessary, ignore eliminated rows or columns. This is useful for color.

BBO_PAL_COLORS
: The color table passed in with the Bitmap-InfoTable is defined as indices into the currently realized palette, rather than actual colors.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiDrawBits - Remarks

A rectangle of bit-map image data is copied from storage to a bit map selected into a device context associated with the target presentation space. Alternatively, the target presentation space can be associated with a device context that specifies a suitable raster device, for example, the screen. An error occurs if this device does not support raster operations.

The source bits must be in one of the standard bit-map formats.

A rectangle is specified in device coordinates for the source bits, and one in world coordinates for the target presentation space. The source rectangle is noninclusive; the left and lower boundaries in device space are included, but not the right and upper boundaries. Thus if the bottom left is equal to the top right, the rectangle is deemed to be empty. The target rectangle is "inclusive-inclusive"; that is, all boundaries are included in the rectangle.

If the target rectangle, after transformation to device coordinates and adjustment for inclusivity, is not the same size as the source rectangle, then stretching or compressing of the data occurs. *flOptions* specifies how eliminated rows or columns of bits are to be treated if compression occurs. Note that the pattern data is never stretched or compressed.

These current attributes of the target presentation space are used (other than for converting between monochrome and color, as described below):

- Area color
- Area background color
- Pattern set
- Pattern symbol.

The color values are used in conversion between monochrome and color data. This is the only format conversion performed by this function. The conversions are:

- Output of a monochrome pattern to a color device

 In this instance the pattern is converted first to a color pattern, using the current area colors:

 - source 1s -> area foreground color
 - source 0s -> area background color.

- Copying from a monochrome bit map to a color bit map (or device)

 The source bits are converted as follows:

 - source 1s -> image foreground color
 - source 0s -> image background color.

- Copying from a color bit map to a monochrome bit map (or device)

 The source bits are converted as follows:

- source nonzeros -> image foreground color
- source 0s -> image background color.

If the mix (*lRop*) does not call for a pattern, the pattern set and pattern symbol are not used.

Neither the source nor the pattern is required when a bit map, or part of a bit map, is to be cleared to a particular color.

If the mix does require both source and pattern, a three-way operation is performed.

If a pattern is required, dithering may be performed for solid patterns in a color that is not available on the device. See GpiSetPattern.

Support for the BM_SRCTRANSPARENT and BM_DESTTRANSPARENT background mix options is provided by the CAPS_BACKGROUND_MIX_SUPPORT flags in DevQueryCaps.

Requesting the BM_SRCTRANSPARENT background mix option results in pels from the source bit map matching the presentaton space background color, to not be copied to the output bit map, effectively leaving those pels in the output unchanged. This provides for a transparent overlay function.

Requesting the BM_DESTTRANSPARENT background mix option results in a transfer such that pels from the source bit map will only be copied to destination pels that match the presentation space background color. This provides for a transparent underlay function.

This function can cause immediate drawing, or be retained in segment store, or both of these, depending upon the drawing mode (see GpiSetDrawingMode). If the function is retained in segment store, the storage identified by the *pBits* and *pbmiInfoTable* parameters must not be changed or freed by the application while the segment containing the function can still be drawn. However, if a metafile is generated and no further drawing is needed, this does not apply, as the information is encaptured in the metafile.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiDrawBits - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_BITBLT_MIX (0x2046)
An invalid *lRop* was specified with a GpiBitBlt or GpiWCBitBlt function.

PMERR_INV_BITBLT_STYLE (0x2047)
An invalid options parameter was specified with a GpiBitBlt or GpiWCBitBlt function.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
An invalid rectangle parameter was specified.

PMERR_INCORRECT_DC_TYPE (0x203C)
An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

GpiDrawBits - Related Functions

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt

GpiDrawBits - Graphic Elements and Orders

Element Type: **OCODE_GBBLT**

Order: Bitblt

GpiDrawBits - Example Code

This example uses GpiDrawBits to draw a rectangle of bits. The bit map was previously placed in application memory using GpiQueryBitmapBits; when the stored image is displayed, it will be a compressed copy (ROP_SRCCOPY) of the source bit map (note the difference between the target and source rectangle sizes), with eliminated rows/columns ignored (BBO_IGNORE) when compression takes place.

```
#define INCL_GPIBITMAPS         /* Bit map functions            */
#include <os2.h>

HPS    hps;             /* presentation-space handle            */
PBYTE        pb;        /* bit-map image data                   */
BITMAPINFO2  pbmi;      /* bit-map information table            */
LONG         lHits;     /* correlation/error indicator          */
LONG         lScan;     /* number of lines scanned              */
/* target and source rectangles */
POINTL       aptlPoints[4]={ 300, 400, 350, 450, 0, 0, 100, 100 };

/* scan and transfer bit map to application storage */
pbmi.cbFix = 16L;
pbmi.cPlanes = 1;
pbmi.cBitCount = 4;
lScan = GpiQueryBitmapBits(hps, 0L, 100L, pb, &pbmi);
  .
  .
  .
/* draw stored rectangle bit map */
lHits = GpiDrawBits(hps, (VOID *)pb, &pbmi, 4L,
                    aptlPoints, ROP_SRCCOPY, BBO_IGNORE);
```

GpiDrawBits - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiDrawChain

GpiDrawChain - Syntax

This function draws the segments that are in the segment chain.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiDrawChain(hps);
```

GpiDrawChain Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDrawChain Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE

Error occurred.

GpiDrawChain - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDrawChain - Remarks

The segments drawn are all of the retained segments that have the ATTR_CHAINED segment attribute (see GpiSetInitialSegmentAttrs), together with all of the unchained segments that are called from them.

The drawing operation is controlled by the calls set by the draw controls (see GpiSetDrawControl), except for the correlate control. If there is not a segment open at the time of the draw, and this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

If a segment is already open at the time of the draw, GpiCloseSegment processing is not performed at the completion of the draw (except that any unclosed path or area is abandoned with an error). In this instance, if the open segment is the last one drawn (and no dynamic segments had to be drawn), attributes and other parameters are in the correct state to continue drawing in any drawing mode.

Dynamic segments are not drawn if they are found while processing the segment chain. However, depending on the setting of DCTL_DYNAMIC (see GpiSetDrawControl), they may be removed before, and drawn after, the operation.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the chain. This can be done by ensuring that the first segment to be drawn does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiDrawChain. The latter method also resets the clip path to no clipping, which may also be necessary.

It is an error to issue this function while any of these brackets are open:

- Area bracket
- Path bracket
- Element bracket.

GpiDrawChain - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)

An attempt was made to issue a function that is invalid in a micro presentation space.

GpiDrawChain - Related Functions

Related Functions

- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiErase
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawControl
- GpiSetDrawingMode
- GpiSetStopDraw

GpiDrawChain - Example Code

This function uses GpiDrawChain to draw the two chained segments.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

BOOL     fSuccess;      /* success indicator                    */
HPS      hps;           /* presentation-space handle            */

/* The chaining attribute is switched on */
GpiSetInitialSegmentAttrs(hps, ATTR_CHAINED, ATTR_ON);

/* two chained segments are defined */
GpiOpenSegment(hps, 1L);
 .
 .
GpiCloseSegment(hps);

GpiOpenSegment(hps, 2L);
 .
 .
GpiCloseSegment(hps);

/* draw the segment chain */
fSuccess = GpiDrawChain(hps);
```

GpiDrawChain - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions

GpiDrawDynamics

GpiDrawDynamics - Syntax

This function redraws the dynamic segments in, or called from, the segment chain.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiDrawDynamics(hps);
```

GpiDrawDynamics Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDrawDynamics Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDrawDynamics - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDrawDynamics - Remarks

Dynamic segments are those segments in the segment chain that have the ATTR_DYNAMIC segment attribute (see GpiSetInitialSegmentAttrs). It is preferable to position dynamic segments at the start of the segment chain.

Dynamic segments can either be drawn with this function, or by setting the DCTL_DYNAMIC draw control (see GpiSetDrawControl), and issuing one of the other GpiDraw... calls.

If there is no range set by a previous GpiRemoveDynamics, all dynamic segments are redrawn by GpiDrawDynamics). However, if GpiRemoveDynamics specified a range in the segment chain, the redraw is restricted to the dynamic segments that are in, or called from, the selected range. (See GpiRemoveDynamics).

Note: The redraw is controlled by the calls set by previous calls to GpiSetDrawControl.

The "stop draw" condition can be set (from another thread) while GpiDrawDynamics is in progress. This is useful in responding to a new position by setting this condition, and then clearing it and redrawing at the new position.

If "Erase before draw" is set ON (see GpiSetDrawControl), the presentation space is erased before the redraw.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segments. This can be done either by ensuring that the first dynamic segment to be drawn does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiDrawDynamics. The latter method also resets the clip path to no clipping, which may also be necessary.

If this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment. In particular, note that during GpiDrawDynamics, the system forces the foreground mix to FM_XOR and the background mix to BM_LEAVEALONE. It may be necessary to set one or both of these before starting to draw.

GpiDrawDynamics - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_FOR_THIS_DC_TYPE (0x2074)
An attempt has been made to issue GpiRemoveDynamics or GpiDrawDynamics to a presentation space associated with a metafile device context.

GpiDrawDynamics - Related Functions

Related Functions

- GpiDrawChain
- GpiDrawFrom
- GpiDrawSegment
- GpiErase
- GpiGetData
- GpiPutData
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawControl
- GpiSetDrawingMode
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetStopDraw

GpiDrawDynamics - Example Code

This example uses GpiDrawDynamics to redraw the two previously defined dynamic chained segments.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

BOOL     fSuccess;      /* success indicator                    */
HPS      hps;           /* presentation-space handle            */

/* The chaining attribute is switched on */
GpiSetInitialSegmentAttrs(hps, ATTR_CHAINED | ATTR_DYNAMIC,
                          ATTR_ON);

/* two dynamic chained segments are defined */
GpiOpenSegment(hps, 1L);
 .
 .
GpiCloseSegment(hps);

GpiOpenSegment(hps, 2L);
 .
 .
GpiCloseSegment(hps);

/* draw the dynamic segment chain */
fSuccess = GpiDrawDynamics(hps);
```

GpiDrawDynamics - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDrawFrom

GpiDrawFrom - Syntax

This function draws a section of the segment chain.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;            /*  Presentation-space handle. */
LONG    lFirstSegment;  /*  First segment to be drawn; it must be greater than 0. */
LONG    lLastSegment;   /*  Last segment to be drawn; it must be greater than 0. */
BOOL    rc;             /*  Success indicator. */

rc = GpiDrawFrom(hps, lFirstSegment, lLastSegment);
```

GpiDrawFrom Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDrawFrom Parameter - lFirstSegment

lFirstSegment (LONG) - input
First segment to be drawn; it must be greater than 0.

GpiDrawFrom Parameter - lLastSegment

lLastSegment (LONG) - input
Last segment to be drawn; it must be greater than 0.

GpiDrawFrom Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDrawFrom - Parameters

hps (HPS) - input
Presentation-space handle.

lFirstSegment (LONG) - input
First segment to be drawn; it must be greater than 0.

lLastSegment (LONG) - input
Last segment to be drawn; it must be greater than 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDrawFrom - Remarks

Drawing starts at the segment identified by *lFirstSegment* and includes all chained segments (those with the ATTR_CHAINED segment attribute, see GpiSetInitialSegmentAttrs), and the segments called from them, up to, and including, the segment identified by *lLastSegment*.

The drawing operation is controlled by the calls set by the draw controls (see GpiSetDrawControl), except for the correlate control.

If there is not a segment open at the time of the draw, and this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

If a segment is already open at the time of the draw, GpiCloseSegment processing is not performed at the completion of the draw (except that any unclosed path or area is terminated with an error). In this instance, if the open segment is the last one drawn (and no dynamic segments had to be drawn), attributes and other parameters are in the correct state to continue drawing in any drawing mode.

Dynamic segments are not drawn if they are found while processing the segment chain. However, depending on the setting of DCTL_DYNAMIC (see GpiSetDrawControl), they may be removed before, and drawn after, the operation. If this happens, then *all* dynamic segments are involved, whether they occur within the range specified or not.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segments. This can be done either by ensuring that the first segment to be drawn does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiDrawFrom. The latter method also resets the clip path to no clipping, which may also be necessary.

It is an error to issue this function while any of these brackets are open:

- Area bracket
- Path bracket
- Element bracket.

If *lFirstSegment* does not exist, or is not in the segment chain, an error is raised. If the *lLastSegment* does not exist, or is not in the chain,

or is chained before the *lFirstSegment* , no error is raised, and processing continues to the end of the chain.

GpiDrawFrom - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_SEG_NOT_CHAINED (0x20FF)
An attempt was made to issue GpiDrawFrom, GpiCorrelateFrom or GpiQuerySegmentPriority for a segment that was not chained.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

GpiDrawFrom - Related Functions

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawSegment
- GpiErase
- GpiGetData
- GpiPutData
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawControl
- GpiSetDrawingMode
- GpiSetStopDraw

GpiDrawFrom - Example Code

This example uses the GpiDrawFrom function to draw all segments in the picture chain between and including the segments 1 and 4.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS hps;                /* presentation space handle            */

GpiDrawFrom(hps, 1L, 4L);
```

GpiDrawFrom - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiDrawSegment

GpiDrawSegment - Syntax

This function draws the specified segment.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lSegment;  /*  Segment to be drawn; it must be greater than 0. */
BOOL    rc;        /*  Success indicator. */

rc = GpiDrawSegment(hps, lSegment);
```

GpiDrawSegment Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiDrawSegment Parameter - lSegment

lSegment (LONG) - input
Segment to be drawn; it must be greater than 0.

GpiDrawSegment Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDrawSegment - Parameters

hps (HPS) - input
Presentation-space handle.

lSegment (LONG) - input
Segment to be drawn; it must be greater than 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiDrawSegment - Remarks

The drawing operation is controlled by the calls set by the draw controls (see GpiSetDrawControl), except for the correlate control.

If there is not a segment open at the time of the draw, and this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment.

If a segment is already open at the time of the draw, GpiCloseSegment processing is not performed at the completion of the draw (except that any unclosed path or area is abandoned with an error). In this instance, if the open segment is the segment specified in *lSegment*, and no dynamic segments had to be drawn, then attributes and other parameters are in the correct state to continue drawing in any drawing mode.

Depending on the setting of DCTL_DYNAMIC (see GpiSetDrawControl), all of the dynamic segments in the chain may be removed before, and drawn after, the specified segment is drawn. (Note that if the specified segment is itself dynamic, it is only drawn in this way.)

This function differs from the other GpiDraw... calls, in that the segment to be drawn need not be a chained segment.

It may be necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segment. This can be done either by ensuring that the segment does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiDrawSegment. The latter method also resets the clip path to no clipping, which may also be necessary.

It is an error to issue this function while any of these brackets are open:

- Area bracket
- Path bracket
- Element bracket.

GpiDrawSegment - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

GpiDrawSegment - Related Functions

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiErase
- GpiErrorSegmentData
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawControl
- GpiSetDrawingMode
- GpiSetStopDraw

GpiDrawSegment - Example Code

This example uses the GpiDrawSegment function to draw segment 4.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiOpenSegment(hps, 4L);           /* open the segment           */
GpiMove(hps, &ptlStart);           /* move to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw triangle              */
GpiCloseSegment(hps);              /* close the segment          */
    .
    .
    .
```

```
GpiDrawSegment(hps, 4L);           /* draw segment #4            */
```

GpiDrawSegment - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiElement

GpiElement - Syntax

This function adds a single element to the current segment.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;      /*  Presentation-space handle. */
LONG     lType;    /*  Type to be associated with the element. */
PSZ      pszDesc;  /*  Element description. */
LONG     lLength;  /*  Length of content data for the element. */
PBYTE    pbData;   /*  Buffer pointer. */
LONG     lHits;    /*  Correlation and error indicators. */

lHits = GpiElement(hps, lType, pszDesc, lLength,
          pbData);
```

GpiElement Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiElement Parameter - lType

lType (LONG) - input
Type to be associated with the element.

Application-defined elements should have type values in the range 0x81xxxxxx through 0xFFxxxxxx so as to avoid conflict with system-generated elements.

GpiElement Parameter - pszDesc

pszDesc (PSZ) - input
Element description.

This is a variable length character string that is recorded with the element.

GpiElement Parameter - lLength

lLength (LONG) - input
Length of content data for the element.

This must not be greater than 64 512 bytes (63KB).

GpiElement Parameter - pbData

pbData (PBYTE) - input
Buffer pointer.

Element content data.

GpiElement Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiElement - Parameters

hps (HPS) - input
Presentation-space handle.

lType (LONG) - input
Type to be associated with the element.

Application-defined elements should have type values in the range 0x81xxxxxx through 0xFFxxxxxx so as to avoid conflict with system-generated elements.

pszDesc (PSZ) - input
Element description.

This is a variable length character string that is recorded with the element.

lLength (LONG) - input
Length of content data for the element.

This must not be greater than 64 512 bytes (63KB).

pbData (PBYTE) - input
Buffer pointer.

Element content data.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiElement - Remarks

The element is stored in the current segment if the drawing mode (see GpiSetDrawingMode) is **retain** or **draw-and-retain**. It is drawn if the drawing mode is **draw** or **draw-and-retain**.

It is an error if the element data contains any begin or end element orders. Similarly, this function is not valid within an element bracket.

Note: No coordinate conversion is performed by this function. The application must ensure that the coordinates within the element are in the correct format for the presentation space (see GpiCreatePS).

GpiElement - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_DATA_TOO_LONG (0x2016)
An attempt was made to transfer more than the maximum permitted amount of data (64512 bytes) using GpiPutData, GpiGetData, or GpiElement.

PMERR_ALREADY_IN_ELEMENT (0x2002)
An attempt was made to begin a new element while an existing element bracket was already open.

GpiElement - Related Functions

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

GpiElement - Example Code

This example uses GpiElement to add a single element to the current segment: an arc starting at the current position, passing through (10,10), and ending at (5,5).

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#define INCL_GPISEGMENTS        /* Segment functions            */
#define INCL_ORDERS             /* Graphical Order Formats      */
#include <os2.h>

LONG     lHits;         /* correlation/error indicator          */
HPS      hps;           /* presentation-space handle            */
LONG     lType;         /* element type                         */
char     pszDesc[4];    /* element description                  */
LONG     lLength;       /* length of element data               */
LORDER   pbData;        /* pointer to element data              */
ORDERL_GCARC lArcPts = {10L,10L,5L,5L}; /* arc points structure */

GpiOpenSegment(hps, 3L);   /* opens segment to receive element  */

/* type is order code for arc at current position (GARC) */
lType = OCODE_GCARC;

/* call the element 'Arc' */
strcpy(pszDesc,"Arc");

/* length of element data */
lLength = sizeof(LORDER);

/* fill element data structure */
```

```
pbData.idCode = OCODE_GCARC;  /* order code: arc at current
                                 position */
pbData.uchLength = sizeof(ORDERL_GCARC);
/* order data contains arc points structure */
memcpy(pbData.uchData, lArcPts, sizeof(ORDERL_GCARC));

/* add element */
lHits = GpiElement(hps, lType, pszDesc, lLength, (BYTE *)&pbData);

GpiCloseSegment(hps);      /* closes segment that received data */
```

GpiElement - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiEndArea

GpiEndArea - Syntax

This function ends the construction of a shaded area.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lHits;  /*  Correlation and error indicators. */

lHits = GpiEndArea(hps);
```

GpiEndArea Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiEndArea Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiEndArea - Parameters

hps (HPS) - input
Presentation-space handle.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiEndArea - Remarks

The construction is started by the GpiBeginArea function. If necessary, a final line is constructed (to the starting point of the last figure) to close the area.

The current position is not changed, unless a closure line has to be drawn, in which case the current position is moved to the end point of the line.

GpiEndArea - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_AREA (0x20DD)
An attempt was made to end an area using GpiEndArea or during segment drawing while not in an area bracket.

PMERR_COORDINATE_OVERFLOW (0x2014)

An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiEndArea - Related Functions

Prerequisite Functions

- GpiBeginArea

Related Functions

- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet

GpiEndArea - Graphic Elements and Orders

Element Type: **OCODE_GEAR**

Order: End Area

GpiEndArea - Example Code

This example uses the GpiEndArea function to end an area bracket. The function draws the area (a triangle) by filling the outline with the current fill pattern.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiBeginArea(hps, BA_NOBOUNDARY | BA_ALTERNATE);
GpiMove(hps, &ptlStart);
GpiPolyLine(hps, 3L, ptlTriangle);
GpiEndArea(hps);
```

GpiEndArea - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiEndElement

GpiEndElement - Syntax

This function terminates an element started by GpiBeginElement.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiEndElement(hps);
```

GpiEndElement Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiEndElement Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiEndElement - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiEndElement - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_ELEMENT (0x20DF)
An attempt was made to end an element using GpiEndElement or during segment drawing while not in an element bracket.

GpiEndElement - Related Functions

Prerequisite Functions

- GpiBeginElement

Related Functions

- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

GpiEndElement - Example Code

This example uses the GpiEndElement function to end an element bracket.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#include <os2.h>

HPS  hps;               /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */
    .
    .
    .
/* begin the element bracket */
GpiBeginElement(hps, 1L, "Triangle");
GpiMove(hps, &ptlStart);           /* move to start point (0, 0) */
GpiPolyLine(hps, 3L, ptlTriangle); /* draw triangle              */
GpiEndElement(hps);                /* end element bracket        */
```

GpiEndElement - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Example Code
Related Functions
Glossary

GpiEndPath

GpiEndPath - Syntax

This function ends the specification of a path started by GpiBeginPath.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiEndPath(hps);
```

GpiEndPath Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiEndPath Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiEndPath - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiEndPath - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_PATH (0x20E1)
An attempt was made to end a path using GpiEndPath or during segment drawing while not in a path bracket.

GpiEndPath - Related Functions

Prerequisite Functions

- GpiBeginPath

Related Functions

- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetClipPath
- GpiSetColor
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMix
- GpiStrokePath

GpiEndPath - Graphic Elements and Orders

Element Type: **OCODE_GEPTH**

Order: End Path

GpiEndPath - Example Code

This example uses the GpiEndPath function to end a path bracket. When the path bracket is ended, a subsequent call to the GpiFillPath function draws and fills the path.

```
#define INCL_GPIPATHS           /* GPI Path functions           */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiBeginPath(hps, 1L);                /* start the path bracket */
GpiMove(hps, &ptlStart);              /* move to starting point */
GpiPolyLine(hps, 2L, ptlTriangle);    /* draw the three sides   */
GpiCloseFigure(hps);                  /* close the triangle     */
GpiEndPath(hps);                      /* end the path bracket   */
GpiFillPath(hps, 1L, FPATH_ALTERNATE); /* draw and fill the path */
```

GpiEndPath - Topics

Select an item:
Syntax

Parameters
Returns
Errors
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiEqualRegion

GpiEqualRegion - Syntax

This function checks whether two regions are identical.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;        /*  Presentation-space handle. */
HRGN    hrgnSrc1;   /*  Handle of first region. */
HRGN    hrgnSrc2;   /*  Handle of second region. */
LONG    lEquality;  /*  Equality and error indicators. */

lEquality = GpiEqualRegion(hps, hrgnSrc1,
              hrgnSrc2);
```

GpiEqualRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

The regions must be owned by the device identified by the currently associated device context.

GpiEqualRegion Parameter - hrgnSrc1

hrgnSrc1 (HRGN) - input
Handle of first region.

GpiEqualRegion Parameter - hrgnSrc2

hrgnSrc2 (HRGN) - input
Handle of second region.

GpiEqualRegion Return Value - lEquality

lEquality (LONG) - returns
Equality and error indicators.

EQRGN_NOTEQUAL
Not equal
EQRGN_EQUAL
Equal
EQRGN_ERROR
Error.

GpiEqualRegion - Parameters

hps (HPS) - input
Presentation-space handle.

The regions must be owned by the device identified by the currently associated device context.

hrgnSrc1 (HRGN) - input
Handle of first region.

hrgnSrc2 (HRGN) - input
Handle of second region.

lEquality (LONG) - returns
Equality and error indicators.

EQRGN_NOTEQUAL
Not equal
EQRGN_EQUAL
Equal
EQRGN_ERROR
Error.

GpiEqualRegion - Remarks

Both regions must be of the same device class. It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiEqualRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiEqualRegion - Related Functions

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

GpiEqualRegion - Example Code

This example uses GpiEqualRegion to create two regions (each consisting of three rectangles), and then compares them for equality.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>

LONG     lEquality;     /* equality/error indicator             */
HPS      hps;           /* presentation-space handle            */
HRGN     hrgnSrc1;      /* handle for first region              */
HRGN     hrgnSrc2;      /* handle for second region             */
RECTL arcl[3] = { 100, 100, 200, 200,        /* 1st rectangle   */
                  150, 150, 250, 250,        /* 2nd rectangle   */
                  200, 200, 300, 300 };      /* 3rd rectangle   */

/* create two identical regions comprising three rectangles each*/
hrgnSrc1 = GpiCreateRegion(hps, 3L, arcl);
hrgnSrc2 = GpiCreateRegion(hps, 3L, arcl);

lEquality = GpiEqualRegion(hps, hrgnSrc1, hrgnSrc2);
```

GpiEqualRegion - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiErase

GpiErase - Syntax

This function clears the output display of the device context associated with the specified presentation space, to the reset color (CLR_BACKGROUND; see GpiSetColor).

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiErase(hps);
```

GpiErase Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiErase Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiErase - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiErase - Remarks

This operation is independent of the draw controls; see GpiSetDrawControl.

The call is subject to all clipping currently in force; that is, clip path, viewing limits, graphics field, clip region, and visible region.

This function does not perform any bounds collection, or correlation.

Note: This function must not be used when creating metafiles conforming to System Application Architecture* (SAA*) guidelines; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiErase - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiErase - Related Functions

Related Functions

- GpiCreateLogColorTable
- GpiSelectPalette
- GpiSetColor
- GpiSetDrawControl

GpiErase - Example Code

This example uses the GpiErase function to clear the display before drawing.

```
#define INCL_GPICONTROL         /* GPI control Functions        */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* start point                      */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiErase(hps);                          /* clear the display */
GpiMove(hps, &ptlStart);                /* draw a triangle   */
GpiPolyLine(hps, 3L, ptlTriangle);
```

GpiErase - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiErrorSegmentData

GpiErrorSegmentData - Syntax

This function returns information about the last error that occurred during a segment drawing operation.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
PLONG    plSegment;  /*  Segment in which the error occurred. */
PLONG    plContext;  /*  Context of the error. */
LONG     lOff;       /*  Position. */

lOff = GpiErrorSegmentData(hps, plSegment,
         plContext);
```

GpiErrorSegmentData Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiErrorSegmentData Parameter - plSegment

plSegment (PLONG) - output
Segment in which the error occurred.

GpiErrorSegmentData Parameter - plContext

plContext (PLONG) - output
Context of the error.

GPIE_SEGMENT
The error occurred while processing the contents of a retained segment.

GPIE_ELEMENT
The error occurred while processing the contents of a GpiElement function.

GPIE_DATA
The error occurred while processing the contents of a GpiPutData function.

GpiErrorSegmentData Return Value - lOff

lOff (LONG) - returns
Position.

This is either the byte offset or the element number, depending on *plContext*:

>=0
Position
GPI_ALTERROR
Error.

GpiErrorSegmentData - Parameters

hps (HPS) - input
Presentation-space handle.

plSegment (PLONG) - output
Segment in which the error occurred.

plContext (PLONG) - output
Context of the error.

GPIE_SEGMENT
The error occurred while processing the contents of a retained segment.

GPIE_ELEMENT
The error occurred while processing the contents of a GpiElement function.

GPIE_DATA
The error occurred while processing the contents of a GpiPutData function.

lOff (LONG) - returns
Position.

This is either the byte offset or the element number, depending on *plContext*:

>=0
Position
GPI_ALTERROR
Error.

GpiErrorSegmentData - Remarks

The information returned is:

- The segment name
- The context
- The byte offset or element number (depending on the context).

The byte offset is returned for these contexts:

- The error occurred within the data of a GpiElement function
- The error occurred within the data of a GpiPutData function.

The element number is returned for the segment context.

GpiErrorSegmentData - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiErrorSegmentData - Related Functions

Related Functions

- GpiDrawChain

- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiElement
- GpiGetData
- GpiPutData
- GpiRemoveDynamics

GpiErrorSegmentData - Example Code

This example uses GpiErrorSegmentData to query the error context and assigns a variable to the returned element number if the context is an element error.

```
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

LONG     lOff;          /* error or offset/element number       */
HPS      hps;           /* presentation-space handle            */
LONG     plSegment;     /* Segment in which the error occurred  */
LONG     plContext;     /* Context of the error                 */
LONG     lElement;      /* element number causing error         */


lOff = GpiErrorSegmentData(hps, &plSegment, &plContext);

if (plContext == GPIE_ELEMENT)
   lElement = lOff;
```

GpiErrorSegmentData - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiExcludeClipRectangle

GpiExcludeClipRectangle - Syntax

This function excludes a rectangle from the clipping region.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;            /*  Presentation-space handle. */
PRECTL    prclRectangle;  /*  Rectangle to be excluded. */
LONG      lComplexity;    /*  Complexity of clipping and error indicators. */

lComplexity = GpiExcludeClipRectangle(hps,
                prclRectangle);
```

GpiExcludeClipRectangle Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiExcludeClipRectangle Parameter - prclRectangle

prclRectangle (PRECTL) - input
Rectangle to be excluded.

The coordinates are world coordinates.

GpiExcludeClipRectangle Return Value - lComplexity

lComplexity (LONG) - returns
Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

The following list the possible values:

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiExcludeClipRectangle - Parameters

hps (HPS) - input
Presentation-space handle.

prclRectangle (PRECTL) - input
Rectangle to be excluded.

The coordinates are world coordinates.

lComplexity (LONG) - returns
Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

The following list the possible values:

RGN_NULL
Null region
RGN_RECT
Rectangular region
RGN_COMPLEX
Complex region
RGN_ERROR
Error.

GpiExcludeClipRectangle - Remarks

The boundaries of the rectangle are considered to be part of the interior, so that a point on the rectangle boundary is clipped (removed).

This function creates a clip region if one does not currently exist. The application is responsible for freeing this (with GpiDestroyRegion) if it subsequently selects another clip region (see GpiSetClipRegion). Any clip region still selected when the device context is closed is automatically freed.

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiExcludeClipRectangle - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
An invalid rectangle parameter was specified.

GpiExcludeClipRectangle - Related Functions

Related Functions

- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipRegion

GpiExcludeClipRectangle - Example Code

This example uses GpiExcludeClipRectangle to exclude a 100x100 rectangle, anchored at (100,100), from the clipping region.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>

LONG  lComplexity;      /* clipping complexity/error return     */
HPS   hps;              /* Presentation-space handle            */
RECTL prclRectangle = {100, 100, 200, 200};/* exclude rectangle */

lComplexity = GpiExcludeClipRectangle(hps, &prclRectangle);
```

GpiExcludeClipRectangle - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiFillPath

GpiFillPath - Syntax

This function draws the interior of a path using the area attributes.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lPath;     /*  Identifier of path whose interior is to be drawn; it must be 1. */
LONG    lOptions;  /*  Fill option. */
LONG    lHits;     /*  Error indicator. */

lHits = GpiFillPath(hps, lPath, lOptions);
```

GpiFillPath Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiFillPath Parameter - lPath

lPath (LONG) - input
Identifier of path whose interior is to be drawn; it must be 1.

GpiFillPath Parameter - lOptions

lOptions (LONG) - input
Fill option.

One value should be selected for each option (unless the default is suitable). These values can be ORed together to determine fill path mode and boundaries.

Fill path mode:

FPATH_ALTERNATE	Fills the path using the alternate rule; see GpiBeginArea (the default).
FPATH_WINDING	Fills the path using the winding rule; see GpiBeginArea. This value must be selected if the path has been modified using GpiModifyPath.

Draw boundary lines:

FPATH_INCL	Includes all boundaries in fill path (the default).
FPATH_EXCL	Excludes bottom and right boundaries from the fill path.

GpiFillPath Return Value - lHits

lHits (LONG) - returns
Error indicator.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiFillPath - Parameters

hps (HPS) - input
Presentation-space handle.

lPath (LONG) - input
Identifier of path whose interior is to be drawn; it must be 1.

lOptions (LONG) - input
Fill option.

One value should be selected for each option (unless the default is suitable). These values can be ORed together to determine fill path mode and boundaries.

Fill path mode:

FPATH_ALTERNATE
: Fills the path using the alternate rule; see GpiBeginArea (the default).

FPATH_WINDING
: Fills the path using the winding rule; see GpiBeginArea. This value must be selected if the path has been modified using GpiModifyPath.

Draw boundary lines:

FPATH_INCL
: Includes all boundaries in fill path (the default).

FPATH_EXCL
: Excludes bottom and right boundaries from the fill path.

lHits (LONG) - returns
Error indicator.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiFillPath - Remarks

Any open figures within the path are closed.

The path is deleted when the interior has been drawn.

The boundaries of the area, as defined by the path, are considered to be part of the interior and are included in the fill.

If the current drawing mode (see GpiSetDrawingMode) is **draw** or **draw-and-retain**, the interior is drawn on the currently associated device. If the drawing mode is **retain**, this function is stored in the current segment, and output occurs when the segment is subsequently drawn in the usual way.

GpiFillPath - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID (0x20AE)
An invalid path identifier parameter was specified.

PMERR_INV_FILL_PATH_OPTIONS (0x2070)
An invalid options parameter was specified with GpiFillPath.

PMERR_PATH_UNKNOWN (0x20EE)
An attempt was made to perform a path function on a path that did not exist.

GpiFillPath - Related Functions

Prerequisite Functions

- GpiBeginPath

Related Functions

- GpiEndPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetClipPath
- GpiSetColor
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMix
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiStrokePath

GpiFillPath - Graphic Elements and Orders

Element Type: **OCODE_GFPTH**
Note that GpiStrokePath also generates this element type.

Order: Fill Path

GpiFillPath - Example Code

This example uses the GpiFillPath function to draw the interior of the given path. The path, an isosceles triangle, is not closed when it is created, so the GpiFillPath function closes it before filling.

```
#define INCL_GPIPATHS           /* GPI Path functions           */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiBeginPath(hps, 1L);                  /* create a path */
GpiMove(hps, &ptlStart);
GpiPolyLine(hps, 3L, ptlTriangle);
GpiEndPath(hps);

GpiFillPath(hps, 1L, FPATH_ALTERNATE);  /* fill the path */
```

GpiFillPath - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiFloodFill

GpiFloodFill - Syntax

This function fills an area bounded by a given color, or while on a given color.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lOptions;  /*  Flood fill options. */
```

```
LONG    lColor;    /*  Color. */
LONG    lHits;     /*  Correlation and error indicators. */

lHits = GpiFloodFill(hps, lOptions, lColor);
```

GpiFloodFill Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiFloodFill Parameter - lOptions

lOptions (LONG) - input
Flood fill options.

This parameter can have one of the following values:

FF_BOUNDARY
: Fills up to the specified color devaldesc.

FF_SURFACE
: Fills while on the specified color.

GpiFloodFill Parameter - lColor

lColor (LONG) - input
Color.

The boundary or surface color, depending on the value of *lOptions* .

This is either a logical color index, or an RGB value, depending on the state of the color table.

GpiFloodFill Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiFloodFill - Parameters

hps (HPS) - input
Presentation-space handle.

lOptions (LONG) - input
Flood fill options.

This parameter can have one of the following values:

FF_BOUNDARY
Fills up to the specified color devaldesc.
FF_SURFACE
Fills while on the specified color.

lColor (LONG) - input
Color.

The boundary or surface color, depending on the value of *lOptions* .

This is either a logical color index, or an RGB value, depending on the state of the color table.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiFloodFill - Remarks

The seed point is current position, which is unchanged by this function.

The area attributes define the fill.

DevQueryCaps (CAPS_RASTER_FLOOD_FILL) indicates whether GpiFloodFill is supported on any particular device.

The results produced by this function are highly device-dependent.

When the drawing mode is draw, if the presentation space is partially obscured by an overlying window an incorrect fill can result. You can get correct results if your application is in the foreground and no windows or dialogs owned by your application are visible during flood-fill operations.

When filling over a pattern or a dithered color, the individual color of each pel is taken into account.

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiFloodFill - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_FUNCTION_NOT_SUPPORTED (0x1641)
The function is not supported.

PMERR_INV_FLOOD_FILL_OPTIONS (0x210D)
Invalid flood fill parameters were specified.

PMERR_INV_IN_AREA (0x2085)
An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

PMERR_INV_IN_PATH (0x208B)
An attempt was made to issue a function invalid inside a path bracket.

PMERR_INV_COLOR_ATTR (0x2053)
An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INSUFFICIENT_MEMORY (0x203E)
The operation terminated through insufficient memory.

PMERR_START_POINT_CLIPPED (0x2113)
The starting point specified for flood fill is outside the current clipping path or region.

PMERR_NO_FILL (0x2114)
No flood fill occurred because either the starting point color was the same as the input color when a boundary fill was requested, or the starting point color was not the same as the input color when a surface fill was requested.

GpiFloodFill - Related Functions

Prerequisite Functions

- GpiBeginArea
- GpiBeginPath
- GpiFillPath
- GpiSetPel

GpiFloodFill - Example Code

This function uses GpiFloodFill to fill an area bounded by a given color, or while on a given color. The example assumes the color table is in index mode; it fills up to the boundary where the color represented by index 1 appears.

```
#define INCL_GPIBITMAPS         /* Bit map functions            */
#include <os2.h>

LONG  lHits;            /* correlation/error indicator          */
HPS   hps;              /* Presentation-space handle            */
LONG  lOptions;         /* flood fill options                   */
LONG  lColor;           /* color                                */

/* fill up to the boundaries of the color */
lOptions = FF_BOUNDARY;

/* use color corresponding to index 1 */
lColor = 1;
```

```
lHits = GpiFloodFill(hps, lOptions, lColor);
```

GpiFloodFill - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiFrameRegion

GpiFrameRegion - Syntax

This function draws a frame inside a region using the current pattern attributes.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
HRGN      hrgn;       /*  Region handle. */
PSIZEL    thickness;  /*  Thickness of frame. */
LONG      lHits;      /*  Correlation and error indicators. */

lHits = GpiFrameRegion(hps, hrgn, thickness);
```

GpiFrameRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiFrameRegion Parameter - hrgn

hrgn (HRGN) - input
Region handle.

GpiFrameRegion Parameter - thickness

thickness (PSIZEL) - input
Thickness of frame.

The width and height of the rectangle, in device coordinates, used to trace the frame. Both the *cx* and *cy* fields must be greater than or equal to zero.

GpiFrameRegion Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiFrameRegion - Parameters

hps (HPS) - input
Presentation-space handle.

hrgn (HRGN) - input
Region handle.

thickness (PSIZEL) - input
Thickness of frame.

The width and height of the rectangle, in device coordinates, used to trace the frame. Both the *cx* and *cy* fields must be greater than or equal to zero.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiFrameRegion - Remarks

The frame is drawn by tracing around the inner boundary of the region with a rectangle of size given by the *thickness* parameter. The edge of the frame includes the pels on the left and bottom boundaries of the region, unless those pels are also on the top and right boundaries, in which case they are excluded.

No part of the frame is drawn outside the region.

The region is assumed to be defined in device coordinates.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiFrameRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiFrameRegion - Example Code

This example uses GpiFrameRegion to draw a frame of width 5 around an existing region.

```
#define INCL_GPIREGIONS         /* Region functions            */
#include <os2.h>

LONG     lHits;         /* correlation/error indicator         */
HPS      hps;           /* presentation-space handle           */
HRGN     hrgn;          /* handle for region                   */
SIZEL    psizlThickness = {5L,5L};
                        /* Thickness of frame                  */
RECTL arcl[3] = { 100, 100, 200, 200,        /* 1st rectangle  */
                  150, 150, 250, 250,        /* 2nd rectangle  */
                  200, 200, 300, 300 };      /* 3rd rectangle  */

/* create a region comprising three rectangles */
hrgn = GpiCreateRegion(hps, 3L, arcl);

lHits = GpiFrameRegion(hps, hrgn, &psizlThickness);
```

GpiFrameRegion - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiFullArc

GpiFullArc - Syntax

This function creates a full arc with its center at the current position.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;           /*  Presentation-space handle. */
LONG     lControl;      /*  Interior and outline control. */
FIXED    fxMultiplier;  /*  Multiplier. */
LONG     lHits;         /*  Correlation and error indicators. */

lHits = GpiFullArc(hps, lControl, fxMultiplier);
```

GpiFullArc Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiFullArc Parameter - lControl

lControl (LONG) - input
Interior and outline control.

Specifies whether the interior of the full arc should be filled, and whether the outline should be drawn. This parameter can have one

of the following values:

DRO_FILL
: Fill interior

DRO_OUTLINE
: Draw outline

DRO_OUTLINEFILL
: Draw outline and fill interior.

GpiFullArc Parameter - fxMultiplier

fxMultiplier (FIXED) - input
Multiplier.

This determines the size of the arc, in relation to an arc with the current arc parameters. The implementation limit of the multiplier is 255.

The value must not be negative.

GpiFullArc Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiFullArc - Parameters

hps (HPS) - input
Presentation-space handle.

lControl (LONG) - input
Interior and outline control.

Specifies whether the interior of the full arc should be filled, and whether the outline should be drawn. This parameter can have one of the following values:

DRO_FILL
: Fill interior

DRO_OUTLINE
: Draw outline

DRO_OUTLINEFILL
: Draw outline and fill interior.

fxMultiplier (FIXED) - input
Multiplier.

This determines the size of the arc, in relation to an arc with the current arc parameters. The implementation limit of the multiplier is 255.

The value must not be negative.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiFullArc - Remarks

The current position is not changed.

The arc parameters determine whether the full arc is drawn clockwise or counterclockwise.

Either the outline of the full arc, or its interior, or both, can be drawn.

If this function appears within an area or path definition, it generates a complete closed figure (DRO_OUTLINE must be specified). It must not occur within any other figure definition.

If correlation is in force, a hit always results if the pick aperture intersects the full arc boundary. However, if the pick aperture lies wholly within the figure, a hit only occurs if the interior is being drawn (DRO_FILL or DRO_OUTLINEFILL).

GpiFullArc - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_ARC_CONTROL (0x2040)
: An invalid control parameter was specified with GpiFullArc.

PMERR_INV_MULTIPLIER (0x20A7)
: An invalid multiplier parameter was specified with GpiPartialArc or GpiFullArc.

GpiFullArc - Related Functions

Related Functions

- GpiPartialArc
- GpiPointArc
- GpiPop
- GpiSetArcParams
- GpiSetAttrMode

- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetCurrentPosition
- GpiSetDefArcParams
- GpiSetDefAttrs
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix

GpiFullArc - Graphic Elements and Orders

Element Type: **OCODE_GCFARC**

Order: Begin Area
This order is generated only if *lControl* is DRO_FILL or DRO_OUTLINEFILL.

Order: Full Arc at Current Position

Order: End Area
This order is generated only if *lControl* is DRO_FILL or DRO_OUTLINEFILL.

GpiFullArc - Example Code

This example uses GpiFullArc to draw five concentric circles. The arc parameters are set before drawing the arc. Only the outline is drawn for the arc.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
SHORT i;                /* loop variable                        */
ARCPARAMS arcp = { 1, 1, 0, 0 }; /* arc parameters structure    */

GpiSetArcParams(hps, &arcp);

for (i = 5; i > 0; i--)
    GpiFullArc(hps,          /* presentation-space handle       */
        DRO_OUTLINE,         /* outline                         */
        MAKEFIXED(i, 0));    /* converts integer to fixed point */
```

GpiFullArc - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code

Related Functions
Graphic Elements and Orders
Glossary

GpiGetData

GpiGetData - Syntax

This function retrieves a buffer of graphic data from the specified segment into the supplied buffer. The data is a list of drawing orders. For details of these, see Graphics Orders

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;       /*  Presentation-space handle. */
LONG     lSegid;    /*  Segment identifier. */
PLONG    plOffset;  /*  Segment offset. */
LONG     lFormat;   /*  Coordinate type required. */
LONG     lLength;   /*  Length of data buffer. */
PBYTE    pbData;    /*  Data buffer. */
LONG     lCount;    /*  Length of returned data. */

lCount = GpiGetData(hps, lSegid, plOffset,
           lFormat, lLength, pbData);
```

GpiGetData Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiGetData Parameter - lSegid

lSegid (LONG) - input
Segment identifier.

It must be greater or equal to 0.

GpiGetData Parameter - plOffset

plOffset (PLONG) - in/out
Segment offset.

A value used to indicate the position in the segment from which data is to be retrieved. It must be set to 0 the first time GpiGetData is called. This indicates that data is to be obtained from the start of the segment. On return, it contains a value that can be used on a subsequent call to continue data retrieval.

The only possible values that can be specified are 0 or the value returned from a previous function.

GpiGetData Parameter - lFormat

lFormat (LONG) - input
Coordinate type required.

DFORM_NOCONV
No coordinate conversion performed.

This is the only value and it is required. That is, the paramater must be set to this value.

GpiGetData Parameter - lLength

lLength (LONG) - input
Length of data buffer.

GpiGetData Parameter - pbData

pbData (PBYTE) - output
Data buffer.

For order formats, see Graphics Orders.

GpiGetData Return Value - lCount

lCount (LONG) - returns
Length of returned data.

>=0
Number of bytes actually returned in *pbData*
GPI_ALTERROR
Error.

GpiGetData - Parameters

hps (HPS) - input
Presentation-space handle.

lSegid (LONG) - input
Segment identifier.

It must be greater or equal to 0.

plOffset (PLONG) - in/out
Segment offset.

A value used to indicate the position in the segment from which data is to be retrieved. It must be set to 0 the first time GpiGetData is called. This indicates that data is to be obtained from the start of the segment. On return, it contains a value that can be used on a subsequent call to continue data retrieval.

The only possible values that can be specified are 0 or the value returned from a previous function.

lFormat (LONG) - input
Coordinate type required.

DFORM_NOCONV
No coordinate conversion performed.

This is the only value and it is required. That is, the paramater must be set to this value.

lLength (LONG) - input
Length of data buffer.

pbData (PBYTE) - output
Data buffer.

For order formats, see Graphics Orders.

lCount (LONG) - returns
Length of returned data.

>=0
Number of bytes actually returned in *pbData*
GPI_ALTERROR
Error.

GpiGetData - Remarks

If the buffer is large enough to contain the data requested, the data is returned and *lCount* is set to show its length.

If the buffer is not large enough, the buffer is filled and *lCount* is set to the length of the buffer. This may mean that there is an incomplete order at the end of the buffer; even so, it is possible to use GpiPutData subsequently, without having to scan the orders in the buffer.

The application can detect when it has been given all the data by checking the *lCount* value. If this is less than the value of *lLength* specified, there is no more data to be returned. If it is equal, there is more data, except in the case where the data just fits in the buffer, which is detected if another GpiGetData function is issued, and a *lCount* of 0 is returned.

No conversion of coordinates is performed for the DFORM_NOCONV value of the control parameter. The coordinates are in the presentation space format.

This function can be issued while there is a segment open, unless the open segment is the segment referenced by this function. If the segment referenced by this function is open, an error occurs.

The segment transform and viewing transform are not returned by this call.

GpiGetData - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
: An invalid segment identifier was specified.

PMERR_INV_SEG_OFFSET (0x20C9)
: An invalid offset parameter was specified with GpiPutData.

PMERR_INV_GETDATA_CONTROL (0x2079)
: An invalid format parameter was specified with GpiGetData.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_SEG_NOT_FOUND (0x2100)
: The specified segment identifier did not exist.

PMERR_SEG_IS_CURRENT (0x20FE)
: An attempt was made to issue GpiGetData to a segment that was currently open.

PMERR_DATA_TOO_LONG (0x2016)
: An attempt was made to transfer more than the maximum permitted amount of data (64512 bytes) using GpiPutData, GpiGetData, or GpiElement.

GpiGetData - Related Functions

Related Functions

- GpiPutData

GpiGetData - Example Code

This example uses the GpiGetData function to copy data from one segment to another.

```
#define INCL_GPISEGMENTS      /* Segment functions              */
#include <os2.h>

HPS hps;                        /* presentation space handle       */
LONG fFormat = DFORM_NOCONV;  /* does not convert coordinates    */
LONG offSegment = 0L;           /* offset in segment               */
LONG offNextElement = 0L;       /* offset in segment to next element */
LONG cb = 0L;                   /* bytes retrieved          */
BYTE abBuffer[512];    /* data buffer                          */
```

```
GpiOpenSegment(hps, 3L);         /* opens segment to receive data   */
do {
    offSegment += cb;
    offNextElement = offSegment;
    cb = GpiGetData(hps, 2L, &offNextElement, fFormat, 512L,
                    abBuffer);

    /* Put data in other segment. */

    if (cb > 0L) GpiPutData(hps, /* presentation-space handle   */
        fFormat,                 /* format of coordinates       */
        &cb,                /* number of bytes in buffer   */
        abBuffer);               /* buffer with graphics-order data */

} while (cb > 0);
GpiCloseSegment(hps);            /* closes segment that received data */
```

GpiGetData - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiImage

GpiImage - Syntax

This function draws a rectangular image, with the top-left corner at the current position.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;             /*  Presentation-space handle. */
LONG      lFormat;         /*  Format of image data. */
PSIZEL    psizlImageSize;  /*  Size of image area (in pels). */
LONG      lLength;         /*  Length in bytes of image data. */
PBYTE     pbData;          /*  Image data. */
LONG      lHits;           /*  Correlation and error indicators. */

lHits = GpiImage(hps, lFormat, psizlImageSize,
          lLength, pbData);
```

GpiImage Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiImage Parameter - lFormat

lFormat (LONG) - input
Format of image data.

This is a reserved field; must be set to 0.

GpiImage Parameter - psizlImageSize

psizlImageSize (PSIZEL) - input
Size of image area (in pels).

The maximum width allowed is 2 040.

GpiImage Parameter - lLength

lLength (LONG) - input
Length in bytes of image data.

It must be greater than 0.

GpiImage Parameter - pbData

pbData (PBYTE) - input
Image data.

GpiImage Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiImage - Parameters

hps (HPS) - input
Presentation-space handle.

lFormat (LONG) - input
Format of image data.

This is a reserved field; must be set to 0.

psizlImageSize (PSIZEL) - input
Size of image area (in pels).

The maximum width allowed is 2 040.

lLength (LONG) - input
Length in bytes of image data.

It must be greater than 0.

pbData (PBYTE) - input
Image data.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiImage - Remarks

All images are a rectangular array of pels (display points), each pel being represented by one bit.

psizlImageSize , which defines the width and height of the image, determines how many pels there are in the horizontal and vertical directions.

pbData determines which of the pels are visible; a 1 bit sets the associated pel, using the image foreground color and mix, and a 0 bit sets the pel using the image background color and mix.

The top left-hand corner of the image is placed at the current position, and the data supplied is drawn row by row, starting at the top. Each row is drawn from left to right and must be padded out to an integral number of bytes if the image width specified is not a multiple of 8. For example, if the image width specified is 12, each row of data must be padded out to a length of 16 so that the data in the row occupies exactly 2 bytes.

Within each byte the high-order bit is drawn on the left.

The length of image data specified must include the padding of each row of data. The length must be given in bytes, and an error message is issued if it is wrong.

If the image is being stored in a metafile, then (((pels_per_row + 9) / 8) * pels_per_column) + 10, must be less than 32768.

Because of the different sizes of pels for different devices, the relationship of the image with respect to other graphics primitives is device-dependent.

The current position remains unchanged after the image has been drawn.

GpiImage - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IMAGE_FORMAT (0x2084)
An invalid *lFormat* parameter was specified with GpiImage.

PMERR_INV_IMAGE_DATA_LENGTH (0x2082)
An invalid *lLength* parameter was specified with GpiImage. There is a mismatch between the image size and the data length.

PMERR_INV_IMAGE_DIMENSION (0x2083)
An invalid *psizlImageSize* parameter was specified with GpiImage.

GpiImage - Related Functions

Related Functions

- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetMix

GpiImage - Graphic Elements and Orders

Element Type: **OCODE_GCBIMG**

Order: Begin Image at Current Position

Order: Image Data
One order for each pel row of the image.

Order: End Image

GpiImage - Example Code

This example uses GpiImage to draw an 8-x-8 image. The image data is specified as an array of bytes.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
SIZEL sizl = { 8, 8 };  /* image is 8 pels wide by 8 pels high  */
BYTE abImage[] = { 0x00, 0x18, 0x3c, 0x7e, 0xff,
    0xff, 0x7e, 0x3c, 0x18, 0x00 }; /* image data               */

GpiImage(hps, 0L, &sizl, 8L, abImage);       /* draws the image */
```

GpiImage - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiIntersectClipRectangle

GpiIntersectClipRectangle - Syntax

This function sets the new clip region to the intersection of the current clip region and the specified rectangle.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;           /*  Presentation-space handle. */
PRECTL    prclRectangle; /*  prclRectangle, the coordinates of which are world coordinates. */
LONG      lComplexity;   /*  Complexity of clipping and error indicators. */

lComplexity = GpiIntersectClipRectangle(hps,
                prclRectangle);
```

GpiIntersectClipRectangle Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiIntersectClipRectangle Parameter - prclRectangle

prclRectangle (PRECTL) - input
prclRectangle , the coordinates of which are world coordinates.

GpiIntersectClipRectangle Return Value - lComplexity

lComplexity (LONG) - returns
Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

The possible values for this parameter are:

RGN_NULL
Null region
RGN_RECT
Rectangular region
RGN_COMPLEX
Complex region
RGN_ERROR
Error.

GpiIntersectClipRectangle - Parameters

hps (HPS) - input
Presentation-space handle.

prclRectangle (PRECTL) - input
prclRectangle , the coordinates of which are world coordinates.

lComplexity (LONG) - returns
Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

The possible values for this parameter are:

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiIntersectClipRectangle - Remarks

The boundaries of the rectangle are considered to be part of the interior, so that a point on the rectangle boundary is not clipped (removed) if it was previously within the clip region.

This function creates a clip region if one does not currently exist. The application is responsible for freeing this (with GpiDestroyRegion), if it subsequently selects another clip region (see GpiSetClipRegion). Any clip region still selected when the device context is closed is automatically freed.

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiIntersectClipRectangle - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
: An invalid rectangle parameter was specified.

GpiIntersectClipRectangle - Related Functions

Related Functions

- GpiExcludeClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion

- GpiSetClipRegion

GpiIntersectClipRectangle - Example Code

This example uses GpiIntersectClipRectangle to create a new clipping region, consisting of the intersection of the old clipping region and a 100x100 rectangle, anchored at (100,100).

```
#define INCL_GPIREGIONS         /* Region functions            */
#include <os2.h>

LONG  lComplexity;      /* clipping complexity/error return    */
HPS   hps;              /* Presentation-space handle           */
RECTL prclRectangle = {100,100,200,200}; /* intersect rectangle */

lComplexity = GpiIntersectClipRectangle(hps, &prclRectangle);
```

GpiIntersectClipRectangle - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiLabel

GpiLabel - Syntax

This function generates an element containing the specified label.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lLabel;  /*  Required label. */
BOOL    rc;      /*  Success indicator. */

rc = GpiLabel(hps, lLabel);
```

GpiLabel Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiLabel Parameter - lLabel

lLabel (LONG) - input
Required label.

No check is made on the value of this parameter.

GpiLabel Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiLabel - Parameters

hps (HPS) - input
Presentation-space handle.

lLabel (LONG) - input
Required label.

No check is made on the value of this parameter.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiLabel - Remarks

This function has no effect unless a retained segment is being constructed. It is invalid within an element bracket. Duplicate labels within a segment are allowed.

GpiLabel - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

GpiLabel - Related Functions

Related Functions

- GpiSetElementPointerAtLabel
- GpiSetTag

GpiLabel - Graphic Elements and Orders

Element Type: **OCODE_GLABL**

Order: Label

GpiLabel - Example Code

This example uses the GpiLabel function to create label elements in a segment. If the segment is subsequently edited, the label elements can still be used to locate the elements near it.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#include <os2.h>

HPS  hps;               /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */
```

```
GpiOpenSegment(hps, 4L);          /* creates a segment */
GpiLabel(hps, 5L);                /* creates label 5   */
GpiLabel(hps, 10L);               /* creates label 10  */
GpiMove(hps, &ptlStart);
GpiCloseSegment(hps);
GpiPolyLine(hps, 3L, ptlTriangle);
```

GpiLabel - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiLine

GpiLine - Syntax

This function draws a straight line from the current position to the specified end point.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hps;           /*  Presentation-space handle. */
PPOINTL    pptlEndPoint;  /*  End point of the line. */
LONG       lHits;         /*  Correlation and error indicators. */

lHits = GpiLine(hps, pptlEndPoint);
```

GpiLine Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiLine Parameter - pptlEndPoint

pptlEndPoint (PPOINTL) - input
End point of the line.

GpiLine Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiLine - Parameters

hps (HPS) - input
Presentation-space handle.

pptlEndPoint (PPOINTL) - input
End point of the line.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiLine - Remarks

The current position is set to the end point of the line.

The line is drawn using the current values of the line color, line mix, line width, and line type attributes.

GpiLine - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)

An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_NESTED_FIGURES (0x20A8)
Nested figures have been detected within a path definition.

GpiLine - Related Functions

Related Functions

- GpiBox
- GpiMove
- GpiPolyLine
- GpiPop
- GpiQueryCurrentPosition
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetCurrentPosition
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMix

GpiLine - Graphic Elements and Orders

Element Type: **OCODE_GCLINE**
Note that GpiPolyLine also generates this element type.

Order: Line at Current Position

GpiLine - Example Code

This example uses GpiLine to draw an X.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
/* point array */
POINTL ptl[4] = { 0, 0, 100, 100, 0, 100, 100, 0 };
```

```
GpiMove(hps, &ptl[0]);
GpiLine(hps, &ptl[1]);
GpiMove(hps, &ptl[2]);
GpiLine(hps, &ptl[3]);
```

GpiLine - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiLoadBitmap

GpiLoadBitmap - Syntax

This function creates and loads a bit map from a resource, and returns the bit-map handle.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hps;       /*  Presentation-space handle. */
HMODULE    Resource;  /*  Resource identity containing the bit map. */
ULONG      idBitmap;  /*  ID of the bit map within the resource file. */
LONG       lWidth;    /*  Width of the bit map in pels. */
LONG       lHeight;   /*  Height of the bit map in pels. */
HBITMAP    hbm;       /*  Bit-map handle. */

hbm = GpiLoadBitmap(hps, Resource, idBitmap,
        lWidth, lHeight);
```

GpiLoadBitmap Parameter - hps

hps (HPS) - input
Presentation-space handle.

The associated device should, if possible, hold the bit map in its own memory. Where this is not possible, main memory is used and the bit map is held in a format compatible with the device.

GpiLoadBitmap Parameter - Resource

Resource (HMODULE) - input
Resource identity containing the bit map.

NULLHANDLE
Use the .EXE file of the application.
Other
Module handle returned from the OS/2 DosLoadModule function.

GpiLoadBitmap Parameter - idBitmap

idBitmap (ULONG) - input
ID of the bit map within the resource file.

This parameter must be less or equal to 0xFFFF.

GpiLoadBitmap Parameter - lWidth

lWidth (LONG) - input
Width of the bit map in pels.

It must be greater or equal to 0.

GpiLoadBitmap Parameter - lHeight

lHeight (LONG) - input
Height of the bit map in pels.

It must be greater or equal to 0.

GpiLoadBitmap Return Value - hbm

hbm (HBITMAP) - returns
Bit-map handle.

<>0
Bit-map handle
GPI_ERROR
Error.

GpiLoadBitmap - Parameters

hps (HPS) - input
Presentation-space handle.

The associated device should, if possible, hold the bit map in its own memory. Where this is not possible, main memory is used and the bit map is held in a format compatible with the device.

Resource (HMODULE) - input
Resource identity containing the bit map.

NULLHANDLE
Use the .EXE file of the application.
Other
Module handle returned from the OS/2 DosLoadModule function.

idBitmap (ULONG) - input
ID of the bit map within the resource file.

This parameter must be less or equal to 0xFFFF.

lWidth (LONG) - input
Width of the bit map in pels.

It must be greater or equal to 0.

lHeight (LONG) - input
Height of the bit map in pels.

It must be greater or equal to 0.

hbm (HBITMAP) - returns
Bit-map handle.

<>0
Bit-map handle
GPI_ERROR
Error.

GpiLoadBitmap - Remarks

Some bit-map functions, including drawing into the bit map, require it to be selected into a memory device context, using GpiSetBitmap. This is true whether device or main memory is used to hold the bit map.

The bit map is stretched to the specified *lWidth* and *lHeight*. If *lWidth* or *lHeight* is 0, the bit map is not stretched in that direction; when, for example, *lWidth* = 0, the bit map is not stretched horizontally, when *lHeight* = 0, it is not stretched vertically.

The bit map may have been created by the icon editor in bit-map mode.

There are a number of standard bit-map formats that should normally be adhered to. Other formats can be used if supported by the device.

The bit map is owned by the process from which this function is issued. It cannot be accessed directly from any other process. If it still exists when the process terminates, it is automatically deleted by the system.

GpiLoadBitmap - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_BITMAP_NOT_FOUND (0x200A)
: A attempt was made to perform a bit-map operation on a bit map that did not exist.

PMERR_INV_BITMAP_DIMENSION (0x2048)
: An invalid dimension was specified with a load bit-map function.

GpiLoadBitmap - Related Functions

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt

GpiLoadBitmap - Example Code

This example uses the GpiLoadBitmap function to load a bit map from the .EXE file into application memory. The bit map is then selected, displayed, and finally, deleted from memory.

```
#define INCL_GPIBITMAPS         /* GPI bit map functions         */
#include <os2.h>

HPS hps;                /* presentation space handle           */
HBITMAP hbm, hbmPrevious;
#define BITMAP_ID 1

/* load the bit map from the EXE */
hbm = GpiLoadBitmap(hps, NULLHANDLE, BITMAP_ID, 100L, 100L);
hbmPrevious = GpiSetBitmap(hps, hbm);  /* select bit map for PS */

/* bit map displayed with GpiBitBlt */

GpiSetBitmap(hps, hbmPrevious);      /* release bit map from PS */
GpiDeleteBitmap(hbm);                /* delete the bit map      */
```

GpiLoadBitmap - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiLoadFonts

GpiLoadFonts - Syntax

This function loads one or more fonts from the specified resource file.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB     hab;          /*  Anchor-block handle. */
PSZ     pszFilename;  /*  Filename. */
BOOL    rc;           /*  Success indicator. */

rc = GpiLoadFonts(hab, pszFilename);
```

GpiLoadFonts Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiLoadFonts Parameter - pszFilename

pszFilename (PSZ) - input
Filename.

This is the fully-qualified name of the font resource.

GpiLoadFonts Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiLoadFonts - Parameters

hab (HAB) - input
Anchor-block handle.

pszFilename (PSZ) - input
Filename.

This is the fully-qualified name of the font resource.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiLoadFonts - Remarks

All of the fonts in the file become available for any presentation space (GPI or VIO) created by the same process. They are not available for any other process.

The format of the font definitions in the resource file is defined in Font-File Format.

When no longer required, the fonts may be unloaded with GpiUnloadFonts.

Note: Fonts loaded with GpiLoadFonts are not available for use for spooled printing, that is if a device type of OD_QUEUED is specified in DevOpenDC; in this case GpiCreateLogFont will never return FONT_MATCH for these fonts. To avoid this, install the fonts as public fonts using the Font Palette object located in the System Setup folder, on both the generating and the receiving workstations if these are different.

GpiLoadFonts - Errors

Possible returns from WinGetLastError

PMERR_INV_FONT_FILE_DATA (0x2073)
The font file specified with GpiLoadFonts, GpiLoadPublicFonts, GpiQueryFontFileDescriptions, or GpiQueryFullFontFileDescs contains invalid data.

GpiLoadFonts - Related Functions

Related Functions

- GpiCreateLogFont
- GpiDeleteSetId
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryKerningPairs
- GpiQueryNumberSetIds
- GpiQuerySetIds
- GpiQueryWidthTable
- GpiSetCharSet
- GpiUnloadFonts

GpiLoadFonts - Example Code

This example uses the GpiLoadFonts function to load all fonts from the font resource file HELV.FON. The GpiQueryFonts function retrieves the number of fonts loaded.

```
#define INCL_GPILCIDS           /* Font functions               */
#include <os2.h>

HPS hps;                /* presentation space handle            */
HAB hab;                /* anchor-block handle                  */
LONG cFonts = 0L;       /* font count                           */
LONG remFonts;          /* fonts not returned                   */

GpiLoadFonts(hab, "C:\\HELV.FON");

remFonts = GpiQueryFonts(hps, QF_PRIVATE, NULL, &cFonts, 0L, NULL);
```

GpiLoadFonts - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiLoadMetaFile

GpiLoadMetaFile - Syntax

This function loads data from a file into a metafile.

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB    hab;          /*  Anchor-block handle. */
PSZ    pszFilename;  /*  Filename. */
HMF    hmf;          /*  Metafile handle or error. */

hmf = GpiLoadMetaFile(hab, pszFilename);
```

GpiLoadMetaFile Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiLoadMetaFile Parameter - pszFilename

pszFilename (PSZ) - input
Filename.

The name of the file that is to be loaded into a metafile.

GpiLoadMetaFile Return Value - hmf

hmf (HMF) - returns
Metafile handle or error.

<>0
Metafile handle
GPI_ERROR
Error.

GpiLoadMetaFile - Parameters

hab (HAB) - input
Anchor-block handle.

pszFilename (PSZ) - input
Filename.

The name of the file that is to be loaded into a metafile.

hmf (HMF) - returns
Metafile handle or error.

<>0
Metafile handle
GPI_ERROR
Error.

GpiLoadMetaFile - Remarks

A metafile is created, into which the data from the file is loaded. The handle of the metafile created is returned in *hmf*; it can be used on subsequent GpiPlayMetaFile or GpiDeleteMetaFile functions.

GpiLoadMetaFile - Errors

Possible returns from WinGetLastError

PMERR_DOSOPEN_FAILURE (0x2024)
A DosOpen call made during GpiLoadMetaFile or GpiSaveMetaFile gave a good return code but the file was not opened successfully.

PMERR_DOSREAD_FAILURE (0x2025)
A DosRead call made during GpiLoadMetaFile gave a good return code. However, it failed to read any more bytes although the file length indicated that there were more to be read.

GpiLoadMetaFile - Related Functions

Related Functions

- GpiCopyMetaFile
- GpiDeleteMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile
- GpiSetMetaFileBits

GpiLoadMetaFile - Example Code

This example uses the GpiLoadMetaFile function to load a metafile with data from the file sample.met. Later, the metafile is deleted by using the GpiDeleteMetaFile function.

```
#define INCL_GPIMETAFILES       /* Metafile functions           */
#include <os2.h>

HAB hab;                /* anchor block handle                  */
HMF hmf;                /* metafile handle                      */

/* loads metafile from disk */
hmf = GpiLoadMetaFile(hab, "sample.met");
    .
    .
    .
GpiDeleteMetaFile(hmf);             /* deletes metafile         */
```

GpiLoadMetaFile - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiLoadPublicFonts

GpiLoadPublicFonts - Syntax

This function loads one or more fonts from the specified resource file, to be available for all applications.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB     hab;          /*  Anchor-block handle. */
PSZ     pszFilename;  /*  Filename. */
BOOL    rc;           /*  Success indicator. */

rc = GpiLoadPublicFonts(hab, pszFilename);
```

GpiLoadPublicFonts Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiLoadPublicFonts Parameter - pszFilename

pszFilename (PSZ) - input
Filename.

This is the fully-qualified name of the font resource. The file-name extension is ".FON".

GpiLoadPublicFonts Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiLoadPublicFonts - Parameters

hab (HAB) - input
Anchor-block handle.

pszFilename (PSZ) - input
Filename.

This is the fully-qualified name of the font resource. The file-name extension is ".FON".

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiLoadPublicFonts - Remarks

All of the fonts in the file become available for any presentation space (GPI or VIO) created by any process.

The format of the font definitions in the resource file is defined in Font-File Format.

Note: Problems can occur when applications load and unload public fonts. See GpiUnloadPublicFonts.

GpiLoadPublicFonts - Errors

Possible returns from WinGetLastError

PMERR_INSUFFICIENT_MEMORY (0x203E)
: The operation terminated through insufficient memory.

PMERR_INV_FONT_FILE_DATA (0x2073)
: The font file specified with GpiLoadFonts, GpiLoadPublicFonts, GpiQueryFontFileDescriptions, or GpiQueryFullFontFileDescs contains invalid data.

GpiLoadPublicFonts - Example Code

This example use GpiLoadPublicFonts to load and make available fonts from a file "TEST.FON", which is assumed to exist in the root directory of the C: drive and is assumed to contain valid fonts.

```
#define INCL_GPILCIDS           /* Font functions               */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HAB   hab;              /* anchor-block handle                  */
char  pszFilename[16];  /* Name of font resource file          */

/* Resource file is named 'TEST.FON' */
strcpy(pszFilename,"C:\\TEST.FON");

fSuccess = GpiLoadPublicFonts(hab, pszFilename);
```

GpiLoadPublicFonts - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiMarker

GpiMarker - Syntax

This function draws a marker with its center at a specified position.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlPoint;  /*  Position of the marker. */
LONG       lHits;      /*  Correlation and error indicators. */

lHits = GpiMarker(hps, pptlPoint);
```

GpiMarker Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiMarker Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Position of the marker.

GpiMarker Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiMarker - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - input
Position of the marker.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiMarker - Remarks

The current position is moved to the specified position. The marker symbol is selected by the current values of the marker set and marker symbol attributes.

GpiMarker - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiMarker - Related Functions

Related Functions

- GpiPolyMarker
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMarker

- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetMix

GpiMarker - Graphic Elements and Orders

Element Type: **OCODE_GMRK**

Note that GpiPolyMarker also generates this element type.

Order: Marker at Given Position

GpiMarker - Example Code

This example uses the GpiMarker function to draw a marker at the point (10,10).

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptl = { 10, 10 }; /* marker point                        */

GpiMarker(hps, &ptl);
```

GpiMarker - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiModifyPath

GpiModifyPath - Syntax

This function modifies the specified path.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lPath;  /*  Path identifier. */
LONG    lMode;  /*  Modification required. */
BOOL    rc;     /*  Success indicator. */

rc = GpiModifyPath(hps, lPath, lMode);
```

GpiModifyPath Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiModifyPath Parameter - lPath

lPath (LONG) - input
Path identifier.

Identifier of the path to be modified; it must be 1.

GpiModifyPath Parameter - lMode

lMode (LONG) - input
Modification required.

This must be:

MPATH_STROKE
Convert the path to one describing the envelope of a wide line.

GpiModifyPath Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE

Successful completion

FALSE
Error occurred.

GpiModifyPath - Parameters

hps (HPS) - input
Presentation-space handle.

lPath (LONG) - input
Path identifier.

Identifier of the path to be modified; it must be 1.

lMode (LONG) - input
Modification required.

This must be:

MPATH_STROKE
Convert the path to one describing the envelope of a wide line.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiModifyPath - Remarks

This function converts the path to one describing the envelope of a wide line stroked using the current geometric wide-line attribute (see GpiSetLineWidthGeom). Note that this and GpiStrokePath are the only calls that can cause geometric wide lines to be constructed.

The envelope includes the effects of line joins, and line ends, according to the current values of these attributes (see GpiSetLineJoin and GpiSetLineEnd). Note these points:

- A line may be joined to an arc, for example. The common point is handled according to the line-join attribute, rather than applying line ends at each end.
- Any open figures within the path are not closed automatically.
- If a figure is closed using GpiCloseFigure, the joining rules are followed, rather than the ending rules, at the start and end point.
- The envelope takes account of any crossings, so that a character such as a stroked "X" does not have a hole in the middle when subsequently drawn in exclusive-OR mode.

After this function, the only calls that can be performed on the path are GpiFillPath, specifying the FPATH_WINDING option, or GpiSetClipPath, specifying the SCP_WINDING option.

GpiModifyPath - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID (0x20AE)
An invalid path identifier parameter was specified.

PMERR_INV_MODIFY_PATH_MODE (0x20A6)
An invalid mode parameter was specified with GpiModifyPath.

PMERR_PATH_UNKNOWN (0x20EE)
An attempt was made to perform a path function on a path that did not exist.

PMERR_COORDINATE_OVERFLOW (0x2014)
An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiModifyPath - Related Functions

Related Functions

- GpiBeginPath
- GpiEndPath
- GpiFillPath
- GpiOutlinePath
- GpiPathToRegion
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetClipPath
- GpiSetColor
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMix
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiStrokePath

GpiModifyPath - Graphic Elements and Orders

Element Type: **OCODE_GMPTH**

Order: Modify Path

GpiModifyPath - Example Code

This example uses the GpiModifyPath function to modify the given path. The GpiFillPath function then draws the path.

```
#define INCL_GPIPATHS           /* GPI Path functions           */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiBeginPath(hps, 1L);                      /* creates path    */
GpiMove(hps, &ptlStart);
GpiPolyLine(hps, 3L, ptlTriangle);
GpiEndPath(hps);

GpiModifyPath(hps,
              1L,
              MPATH_STROKE);     /* modifies path for wide line */
GpiFillPath(hps, 1L, FPATH_ALTERNATE);   /* draws the wide line */
```

GpiModifyPath - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiMove

GpiMove - Syntax

This function moves the current position to the specified point.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlPoint;  /*  Position to which to move. */
BOOL       rc;         /*  Success indicator. */

rc = GpiMove(hps, pptlPoint);
```

GpiMove Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiMove Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Position to which to move.

This position is in world coordinates.

GpiMove Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiMove - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - input
Position to which to move.

This position is in world coordinates.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiMove - Remarks

This function also has the effect of resetting position within a line-type sequence, and, if within an area, of starting a new closed figure and causing any previous one to be closed automatically if necessary.

This function is equivalent to the GpiSetCurrentPosition call, except that, if the current attribute mode is AM_PRESERVE (see GpiSetAttrMode), the current position is **not** saved before being set to a new value by the GpiMove function, and hence cannot be restored using the GpiPop call.

GpiMove - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

GpiMove - Related Functions

Related Functions

- GpiQueryCurrentPosition
- GpiSetCurrentPosition

GpiMove - Graphic Elements and Orders

Element Type: **OCODE_GSCP**
: Note that GpiSetCurrentPosition also generates this element type.

 Order: Set Current Position

GpiMove - Example Code

This example uses the GpiMove function to draw an X. The function moves the current position to the starting point of each leg of the character.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>
```

```
HPS hps;                /* presentation space handle           */
/* point array */
POINTL ptl[4] = { 0, 0, 100, 100, 0, 100, 100, 0 };

GpiMove(hps, &ptl[0]);    /* move to (0,0)   */
GpiLine(hps, &ptl[1]);
GpiMove(hps, &ptl[2]);    /* move to (0,100) */
GpiLine(hps, &ptl[3]);
```

GpiMove - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiOffsetClipRegion

GpiOffsetClipRegion - Syntax

This function moves the clipping region by the specified displacement.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;          /*  Presentation-space handle. */
PPOINTL    pptlPoint;    /*  Displacement. */
LONG       lComplexity;  /*  Complexity of clipping and error indicators. */

lComplexity = GpiOffsetClipRegion(hps, pptlPoint);
```

GpiOffsetClipRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiOffsetClipRegion Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Displacement.

The displacement by which the clipping region is to be moved, expressed as an offset in world coordinates.

GpiOffsetClipRegion Return Value - lComplexity

lComplexity (LONG) - returns
Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

The possbile values for this parameter are:

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiOffsetClipRegion - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - input
Displacement.

The displacement by which the clipping region is to be moved, expressed as an offset in world coordinates.

lComplexity (LONG) - returns
Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

The possbile values for this parameter are:

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiOffsetClipRegion - Remarks

This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiOffsetClipRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_COORDINATE_OVERFLOW (0x2014)
: An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiOffsetClipRegion - Related Functions

Related Functions

- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipRegion

GpiOffsetClipRegion - Example Code

This example uses GpiOffsetClipRegion to move the clipping region right by 3 and up by 3.

```
#define INCL_GPIREGIONS         /* Region functions            */
#include <os2.h>

LONG  lComplexity;      /* clipping complexity/error return    */
HPS   hps;              /* Presentation-space handle           */
POINTL pptlPoint = {3,3}; /* displacement                      */
```

```
lComplexity = GpiOffsetClipRegion(hps, &pptlPoint);
```

GpiOffsetClipRegion - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiOffsetElementPointer

GpiOffsetElementPointer - Syntax

This function sets the element pointer, within the current segment, to the current value plus the specified offset.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;      /*  Presentation-space handle. */
LONG    loffset;  /*  Offset to be added to the element pointer. */
BOOL    rc;       /*  Success indicator. */

rc = GpiOffsetElementPointer(hps, loffset);
```

GpiOffsetElementPointer Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiOffsetElementPointer Parameter - loffset

loffset (LONG) - input
Offset to be added to the element pointer.

GpiOffsetElementPointer Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiOffsetElementPointer - Parameters

hps (HPS) - input
Presentation-space handle.

loffset (LONG) - input
Offset to be added to the element pointer.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiOffsetElementPointer - Remarks

If the resulting value is negative, the element pointer is set to 0. If the resulting value is greater than the number of elements in the segment, it is set to the last element.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress.

This function is invalid within an element bracket.

GpiOffsetElementPointer - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_NO_CURRENT_SEG (0x20E6)
An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

GpiOffsetElementPointer - Related Functions

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer
- GpiSetElementPointerAtLabel

GpiOffsetElementPointer - Example Code

This example uses the GpiOffsetElementPointer function to move to the element associated with a label element. Combining the GpiSetElementPointerAtLabel and GpiOffsetElementPointer functions is a convenient way to locate elements in segments that have been edited.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS  hps;               /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiOpenSegment(hps, 4L);         /* creates a segment with labels */
GpiLabel(hps, 5L);   GpiMove(hps, &ptlStart);
GpiLabel(hps, 10L);  GpiPolyLine(hps, 3L, ptlTriangle);
GpiCloseSegment(hps);
    .
    .
    .
GpiOpenSegment(hps, 4L);
GpiSetElementPointerAtLabel(hps, 10L);/* move to label 10         */
GpiOffsetElementPointer(hps, 1L);     /* move to polyline element */
```

GpiOffsetElementPointer - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiOffsetRegion

GpiOffsetRegion - Syntax

This function moves a region.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
HRGN       Hrgn;        /*  Handle of the region to be moved. */
PPOINTL    pptlOffset;  /*  Offset to be added to the region boundary. */
BOOL       rc;          /*  Success indicator. */

rc = GpiOffsetRegion(hps, Hrgn, pptlOffset);
```

GpiOffsetRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

GpiOffsetRegion Parameter - Hrgn

Hrgn (HRGN) - input
Handle of the region to be moved.

GpiOffsetRegion Parameter - pptlOffset

pptlOffset (PPOINTL) - input
Offset to be added to the region boundary.

GpiOffsetRegion Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiOffsetRegion - Parameters

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

Hrgn (HRGN) - input
Handle of the region to be moved.

pptlOffset (PPOINTL) - input
Offset to be added to the region boundary.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiOffsetRegion - Remarks

This function moves the region to a new position. The new position is obtained by adding the value of *pptlOffset* to all the points that define the region boundary.

An error is raised if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiOffsetRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiOffsetRegion - Related Functions

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

GpiOffsetRegion - Example Code

This example uses GpiOffsetRegion to move a region right by 3 and up by 3.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>

BOOL   fSuccess;        /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
HRGN   Hrgn;            /* handle for region                    */
POINTL pptlOffset = {3,3}; /* displacement                      */

fSuccess = GpiOffsetRegion(hps, Hrgn, &pptlOffset);
```

GpiOffsetRegion - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiOpenSegment

GpiOpenSegment - Syntax

This function opens a segment with the specified identification number.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lSegment;  /*  Segment identifier. */
BOOL    rc;        /*  Success indicator. */

rc = GpiOpenSegment(hps, lSegment);
```

GpiOpenSegment Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiOpenSegment Parameter - lSegment

lSegment (LONG) - input
Segment identifier.

Must be zero or a positive number.

GpiOpenSegment Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiOpenSegment - Parameters

hps (HPS) - input
Presentation-space handle.

lSegment (LONG) - input
Segment identifier.

Must be zero or a positive number.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiOpenSegment - Remarks

A segment is a way of grouping graphics primitives.

If the current drawing mode is **retain** or **draw-and-retain** (see GpiSetDrawingMode), the following occurs:

- If a nonzero identifier is given, and if a segment with the specified identifier does not already exist, a new retained segment is created. If one does already exist, it is reopened in **retain** mode (with the element pointer set to 0), but is an error in **draw-and-retain** mode.
- If an identifier of 0 is given, a new retained segment is created, regardless of whether one with a 0 identifier already exists. There can be more than one segment with an identifier of 0, but such segments can never subsequently be referenced by identifier. When they have been created, they continue to exist until all segments are deleted. Zero segments must be chained and cannot be defined as dynamic.

If the current drawing mode is **draw**, a new nonretained segment is started. No check is made against any possible retained segment identifiers. The current attributes are set to default values (subject to the ATTR_FASTCHAIN segment attribute; see below).

The initial attributes of the segment are as set by GpiSetInitialSegmentAttrs. The attributes may subsequently be changed with GpiSetSegmentAttrs (except for a segment with an identifier of 0). It is an error to try to open a new segment with a drawing mode of **draw** or **draw-and-retain**, with the ATTR_DYNAMIC segment attribute.

This function causes a segment bracket to be started. While the bracket is in effect, any primitive and attribute functions are considered to be part of the segment, and are stored in it if the drawing mode is **retain** or **draw-and-retain**. The bracket is terminated by a GpiCloseSegment. It is an error if GpiOpenSegment is issued when a segment is already open.

The following actions occur when drawing of a chained segment is started (either as it is passed across the API in **draw** or **draw-and-retain**, or as it is found during a draw operation):

- Current attributes and arc parameters are reset to default values.
- The current tag is reset to its default value.

- Current model transform is reset to unity.
- Current position is set to (0,0).
- The current clip path is set so as to cause no clipping.
- The current viewing limits are reset to their default values.
- The current viewing transform is set either to the value last set by GpiSetViewingTransformMatrix, or to the default value if no GpiSetViewingTransformMatrix function has been issued.

If the segment has the ATTR_FASTCHAIN attribute, the application should not depend upon whether or not these operations are performed. This avoids complications when interchanging picture data with other implementations.

Note: The current clip region is not changed by this function.

If any primitive/attribute calls are issued immediately before this function (that is, outside a segment bracket), then any currently open area, path, or element brackets are terminated, as described for GpiCloseSegment, before the new segment is opened.

If the segment being defined is to be called from another segment (see GpiCallSegmentMatrix), ensure that the viewing transform (see GpiSetViewingTransformMatrix) is unity before first opening the segment.

The maximum number of retained segments allowed for a given presentation space at any time is 16378.

GpiOpenSegment - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_ALREADY_IN_SEG (0x2004)
An attempt was made to open a new segment while an existing segment bracket was already open.

PMERR_PATH_INCOMPLETE (0x20EC)
An attempt was made to open or close a segment either directly or during segment drawing, or to issue GpiAssociate while there is an open path bracket.

PMERR_AREA_INCOMPLETE (0x2005)
One of the following has occurred:
o A segment has been opened, closed,
or drawn.
o GpiAssociate was issued while an
area bracket was open.
o A drawn segment has opened an area
bracket and ended without closing it.

PMERR_INV_MODE_FOR_REOPEN_SEG (0x20A5)
An attempt was made to reopen an existing segment while the drawing mode was set to DM_DRAW or DM_DRAWANDRETAIN.

PMERR_DYNAMIC_SEG_ZERO_INV (0x2029)
An attempt was been made to open a dynamic segment with a segment identifier of zero.

PMERR_INV_MODE_FOR_OPEN_DYN (0x20A4)
An attempt was made to open a segment with the ATTR_DYNAMIC segment set, while the drawing mode was set to DM_DRAW or DM_DRAWANDRETAIN.

PMERR_UNCHAINED_SEG_ZERO_INV (0x2108)
An attempt was made to open segment with segment identifier zero and the ATTR_CHAINED segment attribute not specified.

GpiOpenSegment - Related Functions

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiQueryInitialSegmentAttrs
- GpiQuerySegmentAttrs
- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiSetInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority
- GpiSetViewingTransformMatrix

GpiOpenSegment - Example Code

This example uses the GpiOpenSegment to create a new segment. The segment is subsequently drawn by using the GpiDrawSegment function.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiOpenSegment(hps, 1L);       /* opens the segment             */
GpiMove(hps, &ptlStart);       /* moves to starting point (0,0) */
GpiPolyLine(hps, 3L, ptlTriangle);/* draws triangle             */
GpiCloseSegment(hps);          /* closes the segment            */

GpiDrawSegment(hps, 1L);
```

GpiOpenSegment - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiOutlinePath

GpiOutlinePath - Syntax

This function draws the outline of a path.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lPath;     /*  Identifier of path to be outlined; it must be 1. */
LONG    lOptions;  /*  Options. */
LONG    lHits;     /*  Correlation and error indicators. */

lHits = GpiOutlinePath(hps, lPath, lOptions);
```

GpiOutlinePath Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiOutlinePath Parameter - lPath

lPath (LONG) - input
Identifier of path to be outlined; it must be 1.

GpiOutlinePath Parameter - lOptions

lOptions (LONG) - input
Options.

Reserved; must be 0.

GpiOutlinePath Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiOutlinePath - Parameters

hps (HPS) - input
Presentation-space handle.

lPath (LONG) - input
Identifier of path to be outlined; it must be 1.

lOptions (LONG) - input
Options.

Reserved; must be 0.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiOutlinePath - Remarks

The outline of the path is drawn, using the line attributes, including cosmetic line width (see GpiSetLineWidth) but not geometric line width (see GpiSetLineWidthGeom). This normally has the same effect as if the lines, curves, and so on, which comprise the path, had been drawn without defining them as being within a path. However, if character strings (referencing outline fonts) are contained within the path, the outlines of the characters, without the interior fill, are drawn by GpiOutlinePath, giving the appearance of hollow characters.

Open figures within the path are not closed automatically.

When the outline of the path has been drawn, the path is deleted.

GpiOutlinePath - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID (0x20AE)
An invalid path identifier parameter was specified.

PMERR_INV_RESERVED_FIELD (0x20C1)
An invalid reserved field was specified.

PMERR_PATH_UNKNOWN (0x20EE)
An attempt was made to perform a path function on a path that did not exist.

GpiOutlinePath - Related Functions

Related Functions

- GpiBeginPath
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiPathToRegion
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetClipPath
- GpiSetColor
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix
- GpiStrokePath

GpiOutlinePath - Graphic Elements and Orders

Element Type: **OCODE_GOPTH**

Order: Outline Path

GpiOutlinePath - Example Code

This example uses GpiOutlinePath to draw the outline of a path (in this case a triangle).

```
#define INCL_GPIPATHS           /* Path functions               */
#include <os2.h>

LONG   lHits;           /* correlation/error indicator          */
HPS    hps;             /* Presentation-space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiBeginPath(hps, 1L);                /* start the path bracket */
```

```
GpiMove(hps, &ptlStart);               /* move to starting point */
GpiPolyLine(hps, 2L, ptlTriangle);     /* draw the three sides   */
GpiCloseFigure(hps);                   /* close the triangle     */
GpiEndPath(hps);                       /* end the path bracket   */
 .
 .
lHits = GpiOutlinePath(hps, 1L, 0L);
```

GpiOutlinePath - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPaintRegion

GpiPaintRegion - Syntax

This function paints a region into a presentation space, using the current pattern attributes.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
HRGN    hrgn;   /*  Region handle. */
LONG    lHits;  /*  Correlation and error indicators. */

lHits = GpiPaintRegion(hps, hrgn);
```

GpiPaintRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPaintRegion Parameter - hrgn

hrgn (HRGN) - input
Region handle.

GpiPaintRegion Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiPaintRegion - Parameters

hps (HPS) - input
Presentation-space handle.

hrgn (HRGN) - input
Region handle.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiPaintRegion - Remarks

The current GPI area foreground and background colors are used. Mixing is controlled by the area foreground mix only.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

The region is assumed to be defined in device coordinates.

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiPaintRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiPaintRegion - Related Functions

Related Functions

- GpiBeginArea
- GpiBeginPath
- GpiFillPath
- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPop
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiSetRegion

GpiPaintRegion - Example Code

This example uses the GpiPaintRegion function to fill a complex region consisting of three, intersecting rectangles. The region is filled with a red, diagonal pattern.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>
```

```
HPS hps;                /* presentation space handle           */
HRGN hrgn;                /* handle for region */
RECTL arcl[3] = { 100, 100, 200, 200,    /* 1st rectangle     */
    150, 150, 250, 250,                  /* 2nd rectangle     */
    200, 200, 300, 300 };                /* 3rd rectangle     */

hrgn = GpiCreateRegion(hps, 3L, arcl);
GpiSetColor(hps, CLR_RED);
GpiSetPattern(hps, PATSYM_DIAG1);
GpiPaintRegion(hps, hrgn);
```

GpiPaintRegion - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiPartialArc

GpiPartialArc - Syntax

This function draws a straight line, followed by an arc.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;           /*  Presentation-space handle. */
PPOINTL    pptlCenter;    /*  Center point. */
FIXED      fxMultiplier;  /*  Multiplier. */
FIXED      fxStartAngle;  /*  Start angle in degrees. */
FIXED      fxSweepAngle;  /*  Sweep angle in degrees. */
LONG       lHits;         /*  Correlation and error indicators. */

lHits = GpiPartialArc(hps, pptlCenter, fxMultiplier,
          fxStartAngle, fxSweepAngle);
```

GpiPartialArc Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPartialArc Parameter - pptlCenter

pptlCenter (PPOINTL) - input
Center point.

Center of the arc.

GpiPartialArc Parameter - fxMultiplier

fxMultiplier (FIXED) - input
Multiplier.

This determines the size of the arc in relation to an arc with the current arc parameters.

The implementation limit for the multiplier is 255.

The value must *not* be negative.

GpiPartialArc Parameter - fxStartAngle

fxStartAngle (FIXED) - input
Start angle in degrees.

The value must be positive.

GpiPartialArc Parameter - fxSweepAngle

fxSweepAngle (FIXED) - input
Sweep angle in degrees.

The value must be positive.

GpiPartialArc Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPartialArc - Parameters

hps (HPS) - input
Presentation-space handle.

pptlCenter (PPOINTL) - input
Center point.

Center of the arc.

fxMultiplier (FIXED) - input
Multiplier.

This determines the size of the arc in relation to an arc with the current arc parameters.

The implementation limit for the multiplier is 255.

The value must *not* be negative.

fxStartAngle (FIXED) - input
Start angle in degrees.

The value must be positive.

fxSweepAngle (FIXED) - input
Sweep angle in degrees.

The value must be positive.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPartialArc - Remarks

This function draws two figures:

- A straight line, from the current position to the starting point of an arc
- An arc, with its center at the specified point.

The full arc, of which the arc is a part, is identical to that defined by GpiFullArc. The part of the arc drawn by this primitive is defined by the parameters *fxStartAngle* and *fxSweepAngle*, that are the start and sweep angles, subtended from the center, if the current arc parameters specify a circular form. If they do not, these angles are skewed to the same degree that the ellipse is a skewed circle. *fxStartAngle* is

measured counterclockwise from the x axis of the circle before application of the arc parameters. Both angles must be positive; whether the arc is drawn clockwise or counterclockwise is determined by the arc parameters.

Current position is updated to the final point on the arc.

Note: This differs from GpiFullArc, where current position remains at the center of the figure. A primitive (such as GpiLine) following GpiPartialArc draws from the end point of the arc.

A segment of a pie can be drawn by the following calling sequence:

1. GpiMove, to center of pie
2. GpiPartialArc, drawing one spoke and the arc
3. GpiLine, back to center.

The third step can be performed implicitly by autoclosure if an area is being drawn.

A closed figure bounded by a chord and an arc can be drawn by the following calling sequence:

1. GpiSetLineType to invisible
2. GpiPartialArc, with *fxStartAngle* = angle2, and *fxSweepAngle* = 0, to define one end of the chord
3. GpiSetLineType to visible
4. GpiPartialArc, with *fxStartAngle* = angle1, and *fxSweepAngle* = angle2-angle1.

(In the second example, angle2 is greater than angle1. If the interior of the chord is to be shaded, the area must start after step 2 or 3.)

A sweep angle of greater than 360 degrees is valid, and means that after the initial line a full arc is drawn, followed by a partial arc with a sweep angle of (*fxSweepAngle* MOD 360) degrees.

GpiPartialArc - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MULTIPLIER (0x20A7)
An invalid multiplier parameter was specified with GpiPartialArc or GpiFullArc.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_ANGLE_PARM (0x203F)
An invalid angle parameter was specified with GpiPartialArc.

PMERR_INV_NESTED_FIGURES (0x20A8)
Nested figures have been detected within a path definition.

GpiPartialArc - Related Functions

Related Functions

- GpiFullArc
- GpiPointArc
- GpiPop
- GpiSetAttrMode
- GpiSetArcParams
- GpiSetAttrs
- GpiSetBackColor

- GpiSetBackMix
- GpiSetColor
- GpiSetDefArcParams
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMix

GpiPartialArc - Graphic Elements and Orders

Element Type: **OCODE_GCPARC**

Order: Partial Arc at Current Position

GpiPartialArc - Example Code

This example uses the GpiPartialArc function to draw a chord (an arc whose end points are connected by a straight line).

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptl = { 100, 100 };             /* center point for arc */

GpiSetLineType(hps, LINETYPE_INVISIBLE);
GpiPartialArc(hps, &ptl, MAKEFIXED(50, 0), MAKEFIXED(0, 0),
    MAKEFIXED(180, 0));
GpiSetLineType(hps, LINETYPE_SOLID);
GpiPartialArc(hps, &ptl, MAKEFIXED(50, 0), MAKEFIXED(0, 0),
    MAKEFIXED(180, 0));
```

GpiPartialArc - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPathToRegion

GpiPathToRegion - Syntax

This function converts a path to a region.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     GpiH;      /*  Presentation-space handle. */
LONG    lPath;     /*  Identifier of path to be converted; it must be 1. */
LONG    lOptions;  /*  Fill options. */
HRGN    hrgn;      /*  Region handle. */

hrgn = GpiPathToRegion(GpiH, lPath, lOptions);
```

GpiPathToRegion Parameter - GpiH

GpiH (HPS) - input
Presentation-space handle.

GpiPathToRegion Parameter - lPath

lPath (LONG) - input
Identifier of path to be converted; it must be 1.

GpiPathToRegion Parameter - lOptions

lOptions (LONG) - input
Fill options.

This parameter can have one of the following values:

FPATH_ALTERNATE
Fills the path using the alternate rule; see GpiBeginArea.

This is the default value.

FPATH_WINDING
Fills the path using the winding rule; see GpiBeginArea. This value must be selected if the path has been modified using GpiModifyPath.

GpiPathToRegion Return Value - hrgn

hrgn (HRGN) - returns
Region handle.

<>0
Region handle
RGN_ERROR
Error.

GpiPathToRegion - Parameters

GpiH (HPS) - input
Presentation-space handle.

lPath (LONG) - input
Identifier of path to be converted; it must be 1.

lOptions (LONG) - input
Fill options.

This parameter can have one of the following values:

FPATH_ALTERNATE
Fills the path using the alternate rule; see GpiBeginArea.

This is the default value.

FPATH_WINDING
Fills the path using the winding rule; see GpiBeginArea. This value must be selected if the path has been modified using GpiModifyPath.

hrgn (HRGN) - returns
Region handle.

<>0
Region handle
RGN_ERROR
Error.

GpiPathToRegion - Remarks

This function converts a path (originally defined by a series of GPI drawing calls) to a region. The new region can be operated on by the GPI region calls; in particular GpiCombineRegion can be used to combine it with another region.

This function should not be used in retain or draw-and-retain modes because it does not cause graphics orders to be added to the current segment.

Any open figures within the path are closed automatically.

The boundaries of the area defined by the path are considered to be part of the interior, so that a point on the boundary is included in the new region.

After a path is converted to a region, it no longer exists as a path. The path cannot be reused for any other purpose.

GpiPathToRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID (0x20AE)
: An invalid path identifier parameter was specified.

PMERR_PATH_UNKNOWN (0x20EE)
: An attempt was made to perform a path function on a path that did not exist.

GpiPathToRegion - Related Functions

Related Functions

- GpiBeginPath
- GpiCombineRegion
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiSetClipPath
- GpiStrokePath

GpiPathToRegion - Example Code

This example uses GpiPathToRegion to convert a path (a triangle) to a region using the winding rule to fill the region.

```
#define INCL_GPIPATHS           /* Path functions               */
#include <os2.h>

HRGN   hrgn;            /* handle for region                    */
HPS    hps;             /* Presentation-space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */

GpiBeginPath(hps, 1L);                /* start the path bracket */
GpiMove(hps, &ptlStart);              /* move to starting point */
GpiPolyLine(hps, 2L, ptlTriangle);    /* draw the three sides   */
GpiCloseFigure(hps);                  /* close the triangle     */
GpiEndPath(hps);                      /* end the path bracket   */

hrgn = GpiPathToRegion(hps, 1L, FPATH_WINDING);
```

GpiPathToRegion - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiPlayMetaFile

GpiPlayMetaFile - Syntax

This function plays a metafile into a presentation space.

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;          /*  Presentation-space handle. */
HMF      hmf;          /*  Metafile handle. */
LONG     lCount1;      /*  Count of elements in alOptarray. */
PLONG    alOptarray;   /*  Array of options for playing. */
PLONG    plSegCount;   /*  Reserved value, must be 0. */
LONG     lCount2;      /*  Count of bytes in pszDesc. */
PSZ      pszDesc;      /*  Descriptive record. */
LONG     lHits;        /*  Correlation and error indicators. */

lHits = GpiPlayMetaFile(hps, hmf, lCount1,
          alOptarray, plSegCount, lCount2,
          pszDesc);
```

GpiPlayMetaFile Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPlayMetaFile Parameter - hmf

hmf (HMF) - input
Metafile handle.

Handle of the metafile containing the data.

GpiPlayMetaFile Parameter - lCount1

lCount1 (LONG) - input
Count of elements in *alOptarray* .

It must be greater or equal to 0.

GpiPlayMetaFile Parameter - alOptarray

alOptarray (PLONG) - input
Array of options for playing.

The values of the elements in this array determine what action is to be taken when the metafile is played into the specified presentation space. The elements in the array are numbered consecutively, starting with PMF_SEGBASE. The element number constants start with 0. Any elements in the array that are not set to one of the values defined below must be set to 0.

alOptarray[PMF_SEGBASE]
Reserved; must be 0.

alOptarray[PMF_LOADTYPE]
Specifies what transformations should be performed on the imported picture. The options are:

LT_DEFAULT	The default; same as LT_NOMODIFY
LT_NOMODIFY	The graphics are restored using the current viewing transform (see GpiSetViewingTransformMatrix), rather than the ones that were in use when the data was created. This is the default action. Any change to the graphics field or default viewing transform during the course of the picture will be ignored if this option is specified (or defaulted).
LT_ORIGINALVIEW	The graphics are restored using the viewing transforms that are in the metafile. The default viewing transform of the presentation space is not altered (unless RES_RESET is specified). However, any changes to the default viewing transform that occur during the course of the picture (and also any graphics field clipping) cause changes to the values in the presentation space.

alOptarray[PMF_RESOLVE]
Reserved; must be 0.

alOptarray[PMF_LCIDS]
Specifies the action to be taken for any logical font definitions, or bit maps referenced by local identifiers for use as shading patterns that are held in the metafile.

The options are:

LC_DEFAULT	Default; same as LC_NOLOAD.

LC_NOLOAD	Do not load such objects. This is the default, and is used where the application expects the correct objects to be already loaded.
LC_LOADDISC	Load all objects referenced in the metafile, first deleting any already existing in the presentation space, for which the referenced local identifier is already in use.

alOptarray[PMF_RESET]

Specifies whether the presentation space should be reset before playing the metafile, with the page units and size being set as defined in the metafile.

The options are:

RES_DEFAULT	Default; same as RES_NORESET.
RES_NORESET	Do not perform a reset.
RES_RESET	Reset the presentation space, before loading any logical fonts, color tables, segments, and so on, as follows: 1. Reset the page units and page size to the values contained in the metafile. 2. Set up default transformations, based on the page units and size, as if the presentation space had just been created with these values. 3. Further modify the device transform to ensure that the physical size of the metafile picture is preserved. (Only performed if the page units in the metafile are not PU_ARBITRARY or PU_PELS.) 4. Perform the equivalent of GpiResetPS (option GRES_ALL). 5. Set the default viewing transform to the value specified in the metafile. This option should normally be used with a PMF_LOADTYPE option of LT_ORIGINALVIEW and LC_LOADDISC, but this is not enforced.

alOptarray[PMF_SUPPRESS]

Specifies whether the playing of this metafile actually occurs. This is provided to allow an application to use the PMF_RESET option, and then to regain control to perform further presentation-space modifications if necessary, before playing the remainder of the metafile.

The options are:

SUP_DEFAULT	Default; same as SUP_NOSUPPRESS.
SUP_NOSUPPRESS	Do not suppress the remainder of the metafile.
SUP_SUPPRESS	Suppress the remainder of the metafile. If this option is selected, only processing as determined by the PMF_RESET option is performed. The remainder of the metafile, and all other options, are ignored.

alOptarray[PMF_COLORTABLES]

Specifies the action to be taken with respect to any color table or palette implied or present within the metafile.

The options are:

CTAB_DEFAULT	Default; same as CTAB_NOMODIFY.
CTAB_NOMODIFY	Ignore. The default or loaded color table or

selected palette in the presentation space is unchanged, as are the references to color attributes in the new data. This is the default; it is suitable where it is known that the currently loaded color table or selected palette (if any) is suitable for the use of color in the imported picture.

CTAB_REPLACE — Overwrite the currently-loaded color table (if any), with a color table as implied or present in the metafile. This can be used where there is no existing picture.

CTAB_REPLACEPALETTE — Overwrite the currently-selected palette (if any), with a palette as implied or present in the metafile. This can be used where there is no existing picture.

Note: If the metafile specifies a color table in RGB mode, the currently-selected palette (if any) is overwritten with a color table in RGB mode, and a warning is issued.

alOptarray[PMF_COLORREALIZABLE]
This index/value is maintained for compatibility with 16-bit programs. It is not applicable to 32-bit programming.

alOptarray[PMF_DEFAULTS]
Specifies how the drawing defaults contained in the metafile should be used (see GpiSetDefAttrs, GpiSetDefViewingLimits, GpiSetDefTag, and GpiSetDefArcParams).

The options are:

DDEF_DEFAULT — Default; same as DDEF_IGNORE

DDEF_IGNORE — Ignore any drawing default values in the metafile.

DDEF_LOADDISC — Change any drawing default values in the presentation space that are specified in the metafile, to the values contained in the metafile.

GpiPlayMetaFile Parameter - plSegCount

plSegCount (PLONG) - output
Reserved value, must be 0.

GpiPlayMetaFile Parameter - lCount2

lCount2 (LONG) - input
Count of bytes in *pszDesc* .

It must be greater than 0.

GpiPlayMetaFile Parameter - pszDesc

pszDesc (PSZ) - output
Descriptive record.

pszDesc is a buffer that, on return, contains the descriptive record, of up to 253 bytes, that is saved when the metafile is created (see DevOpenDC in the *Presentation Manager Programming Reference* .) This is null-terminated, even if it has to be truncated.

GpiPlayMetaFile Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiPlayMetaFile - Parameters

hps (HPS) - input
Presentation-space handle.

hmf (HMF) - input
Metafile handle.

Handle of the metafile containing the data.

lCount1 (LONG) - input
Count of elements in *alOptarray* .

It must be greater or equal to 0.

alOptarray (PLONG) - input
Array of options for playing.

The values of the elements in this array determine what action is to be taken when the metafile is played into the specified presentation space. The elements in the array are numbered consecutively, starting with PMF_SEGBASE. The element number constants start with 0. Any elements in the array that are not set to one of the values defined below must be set to 0.

alOptarray[PMF_SEGBASE]
Reserved; must be 0.

alOptarray[PMF_LOADTYPE]
Specifies what transformations should be performed on the imported picture. The options are:

LT_DEFAULT	The default; same as LT_NOMODIFY
LT_NOMODIFY	The graphics are restored using the current viewing transform (see GpiSetViewingTransformMatrix), rather than the ones that were in use when the data was created. This is the default action. Any change to the graphics field or default

viewing transform during the course of the picture will be ignored if this option is specified (or defaulted).

LT_ORIGINALVIEW

The graphics are restored using the viewing transforms that are in the metafile.

The default viewing transform of the presentation space is not altered (unless RES_RESET is specified). However, any changes to the default viewing transform that occur during the course of the picture (and also any graphics field clipping) cause changes to the values in the presentation space.

alOptarray[PMF_RESOLVE]

Reserved; must be 0.

alOptarray[PMF_LCIDS]

Specifies the action to be taken for any logical font definitions, or bit maps referenced by local identifiers for use as shading patterns that are held in the metafile.

The options are:

LC_DEFAULT

Default; same as LC_NOLOAD.

LC_NOLOAD

Do not load such objects. This is the default, and is used where the application expects the correct objects to be already loaded.

LC_LOADDISC

Load all objects referenced in the metafile, first deleting any already existing in the presentation space, for which the referenced local identifier is already in use.

alOptarray[PMF_RESET]

Specifies whether the presentation space should be reset before playing the metafile, with the page units and size being set as defined in the metafile.

The options are:

RES_DEFAULT

Default; same as RES_NORESET.

RES_NORESET

Do not perform a reset.

RES_RESET

Reset the presentation space, before loading any logical fonts, color tables, segments, and so on, as follows:

1. Reset the page units and page size to the values contained in the metafile.
2. Set up default transformations, based on the page units and size, as if the presentation space had just been created with these values.
3. Further modify the device transform to ensure that the physical size of the metafile picture is preserved. (Only performed if the page units in the metafile are not PU_ARBITRARY or PU_PELS.)
4. Perform the equivalent of GpiResetPS (option GRES_ALL).
5. Set the default viewing transform to the value specified in the metafile.

This option should normally be used with a PMF_LOADTYPE option of LT_ORIGINALVIEW and LC_LOADDISC, but this is not enforced.

alOptarray[PMF_SUPPRESS]

Specifies whether the playing of this metafile actually occurs. This is provided to allow an application to use the PMF_RESET option, and then to regain control to perform further presentation-space modifications if necessary, before playing the remainder of the metafile.

The options are:

SUP_DEFAULT	Default; same as SUP_NOSUPPRESS.
SUP_NOSUPPRESS	Do not suppress the remainder of the metafile.
SUP_SUPPRESS	Suppress the remainder of the metafile. If this option is selected, only processing as determined by the PMF_RESET option is performed. The remainder of the metafile, and all other options, are ignored.

alOptarray[PMF_COLORTABLES]
Specifies the action to be taken with respect to any color table or palette implied or present within the metafile.

The options are:

CTAB_DEFAULT	Default; same as CTAB_NOMODIFY.
CTAB_NOMODIFY	Ignore. The default or loaded color table or selected palette in the presentation space is unchanged, as are the references to color attributes in the new data. This is the default; it is suitable where it is known that the currently loaded color table or selected palette (if any) is suitable for the use of color in the imported picture.
CTAB_REPLACE	Overwrite the currently-loaded color table (if any), with a color table as implied or present in the metafile. This can be used where there is no existing picture.
CTAB_REPLACEPALETTE	Overwrite the currently-selected palette (if any), with a palette as implied or present in the metafile. This can be used where there is no existing picture. **Note:** If the metafile specifies a color table in RGB mode, the currently-selected palette (if any) is overwritten with a color table in RGB mode, and a warning is issued.

alOptarray[PMF_COLORREALIZABLE]
This index/value is maintained for compatibility with 16-bit programs. It is not applicable to 32-bit programming.

alOptarray[PMF_DEFAULTS]
Specifies how the drawing defaults contained in the metafile should be used (see GpiSetDefAttrs, GpiSetDefViewingLimits, GpiSetDefTag, and GpiSetDefArcParams).

The options are:

DDEF_DEFAULT	Default; same as DDEF_IGNORE
DDEF_IGNORE	Ignore any drawing default values in the metafile.
DDEF_LOADDISC	Change any drawing default values in the presentation space that are specified in the metafile, to the values contained in the metafile.

plSegCount (PLONG) - output
Reserved value, must be 0.

lCount2 (LONG) - input
Count of bytes in *pszDesc*.

It must be greater than 0.

pszDesc (PSZ) - output
Descriptive record.

pszDesc is a buffer that, on return, contains the descriptive record, of up to 253 bytes, that is saved when the metafile is created (see DevOpenDC in the *Presentation Manager Programming Reference* .) This is null-terminated, even if it has to be truncated.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiPlayMetaFile - Remarks

This function executes the contents of a metafile. This process is known as "playing" the metafile. Whether the graphics are drawn, or retained in segment store, or both, depends upon the current drawing mode (see GpiSetDrawingMode) in the presentation space, for the chained and unchained segment contexts, as appropriate. If chained segments are retained, they are added to the end of any existing segment chain. An error is raised if a segment is to be retained, and it has the same (nonzero) identifier as a currently existing segment.

A segment must not be open when this function is issued. At the completion of the call, there is no open segment.

The application may need to reset the presentation space by GpiResetPS, before issuing this function. Alternatively, the PMF_RESET option on this function may be suitable.

Segments retain the segment attributes that they originally possessed.

GpiPlayMetaFile - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HMF (0x207E)
An invalid metafile handle was specified.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_PLAY_METAFILE_OPTION (0x20B8)
An invalid option parameter was specified with GpiPlayMetaFile.

PMERR_INCOMPATIBLE_METAFILE (0x203B)
An attempt was made to associate a presentation space and a metafile device context with incompatible page units, size or coordinate format; or to play a metafile using the RES_RESET option (to reset the presentation space) to a presentation space that is itself associated with a metafile device context.

PMERR_INV_METAFILE (0x209D)
An invalid metafile was specified with GpiPlayMetaFile.

PMERR_INV_MICROPS_ORDER (0x20A2)
An attempt was made to play a metafile containing orders that are invalid in a micro presentation space.

PMERR_STOP_DRAW_OCCURRED (0x2105)
Segment drawing or GpiPlayMetaFile was stopped prematurely in response to a GpiSetStopDraw request.

PMERR_INV_OUTSIDE_DRAW_MODE (0x20AC)
An attempt was made to issue a GpiSavePS or GpiRestorePS function, or an output only function (for example, GpiPaintRegion) from GpiPlayMetaFile without the drawing mode set to DM_DRAW.

PMERR_INV_ELEMENT_POINTER (0x206B)
An attempt was made to issue GpiPutData with the element pointer not pointing at the last element.

PMERR_INV_IN_CURRENT_EDIT_MODE (0x2087)
An attempt was made to issue a function invalid inside the current editing mode.

PMERR_PROLOG_ERROR (0x20F2)
A prolog error was detected during drawing. Segment prologs are used internally within retained segments and also appear in metafiles. This error can also arise from an End Prolog order that is outside a prolog.

PMERR_DUP_SEG (0x2027)
During GpiPlayMetaFile, while the actual drawing mode was **draw-and-retain** or **retain**, a metafile segment to be stored in the presentation space was found to have the same segment identifier as an existing segment.

GpiPlayMetaFile - Related Functions

Related Functions

- GpiCopyMetaFile
- GpiDeleteMetaFile
- GpiLoadMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile
- GpiSetMetaFileBits

GpiPlayMetaFile - Example Code

This example uses the GpiPlayMetaFile function to play (execute) the metafile loaded by GpiLoadMetaFile into a presentation space associated with a window. GpiPlayMetaFile is called twice: the first call resets the presentation space (by combining the RES_RESET and SUP_SUPPRESS flags), while the second call actually executes the metafile.

```
#define INCL_GPIMETAFILES       /* Metafile functions           */
#define INCL_GPICONTROL         /* GPI control Functions        */
#include <os2.h>

HAB    hab;             /* anchor-block handle                 */
HPS    hps;             /* presentation space handle           */
HMF    hmf;             /* metafile handle                     */
HDC    hdc;             /* Device-context handle               */
HWND   hwnd;            /* window handle                       */
SIZEL  sizl={0,0};      /* use same page size as device        */
CHAR   szBuffer[80];    /* descriptive record buffer           */
LONG     lHits;         /* correlation/error indicator         */
/* play metafile options array */
LONG optArray[PMF_DEFAULTS+1] =
                         {0,LT_DEFAULT,0,LC_DEFAULT,RES_RESET,
                         SUP_SUPPRESS,CTAB_DEFAULT,CREA_DEFAULT,
                         DDEF_DEFAULT};

hmf = GpiLoadMetaFile(hab, "sample.met");

/* create window device context and presentation space,
   associating DC with the PS */
hdc = WinOpenWindowDC(hwnd);
```

```
hps = GpiCreatePS(hab, hdc, &sizl, PU_PELS | GPIA_ASSOC);

/* reset presentation space */
lHits = GpiPlayMetaFile(hps, hmf, 9L, optArray, (LONG *)0, 80L,
                        szBuffer);

/* display metafile in window (reset and
   suppress flags must be changed) */
optArray[PMF_SUPPRESS]=SUP_DEFAULT;
optArray[PMF_RESET]=RES_DEFAULT;
lHits = GpiPlayMetaFile(hps, hmf, 9L, optArray, (LONG *)0, 80L,
                        szBuffer);
```

GpiPlayMetaFile - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiPointArc

GpiPointArc - Syntax

This function creates an arc, using the current arc parameters, through three points, starting at the current position.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;    /*  Presentation-space handle. */
PPOINTL    pptl2;  /*  Intermediate and end points. */
LONG       lHits;  /*  Correlation and error indicators. */

lHits = GpiPointArc(hps, pptl2);
```

GpiPointArc Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPointArc Parameter - pptl2

pptl2 (PPOINTL) - input
Intermediate and end points.

Pointer to an array containing the intermediate and end points.

GpiPointArc Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPointArc - Parameters

hps (HPS) - input
Presentation-space handle.

pptl2 (PPOINTL) - input
Intermediate and end points.

Pointer to an array containing the intermediate and end points.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPointArc - Remarks

The first element of the *pptl2* array defines an intermediate point along the arc, and the second element identifies the end point of the arc. Upon completion, current position is set to the end point of the arc.

GpiPointArc - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_NESTED_FIGURES (0x20A8)
Nested figures have been detected within a path definition.

GpiPointArc - Related Functions

Related Functions

- GpiFullArc
- GpiPartialArc
- GpiPop
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefArcParams
- GpiSetDefAttrs
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix

GpiPointArc - Graphic Elements and Orders

Element Type: **OCODE_GCARC**

Order: Arc at Current Position

GpiPointArc - Example Code

This example uses the GpiPointArc function to draw an arc through the three points of a triangle. The GpiPolyLine function then draws the triangle.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlTriangle[] = { 0, 0, 100, 100, 200, 0 };

GpiMove(hps, &ptlTriangle[0]);    /* moves to start point (0, 0)*/
GpiPointArc(hps, &ptlTriangle[1]);/* draws the arc               */
GpiMove(hps, &ptlTriangle[0]);    /* moves to start point (0, 0)*/
/* draws the triangle */
GpiPolyLine(hps, 3L, &ptlTriangle[1]);
```

GpiPointArc - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPolyFillet

GpiPolyFillet - Syntax

This function draws a curve starting at the current position and defined by the points supplied.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lCount;      /*  Number of points. */
PPOINTL    aptlPoints;  /*  Array of points. */
LONG       lHits;       /*  Correlation and error indicators. */

lHits = GpiPolyFillet(hps, lCount, aptlPoints);
```

GpiPolyFillet Parameter - hps

hps (HPS) - input

Presentation-space handle.

GpiPolyFillet Parameter - lCount

lCount (LONG) - input
Number of points.

Must not be negative. Zero is valid but causes no output.

GpiPolyFillet Parameter - aptlPoints

aptlPoints (PPOINTL) - input
Array of points.

GpiPolyFillet Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiPolyFillet - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of points.

Must not be negative. Zero is valid but causes no output.

aptlPoints (PPOINTL) - input
Array of points.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPolyFillet - Remarks

If two points are supplied, an imaginary straight line is drawn from the current position to the first point and a second straight line from the first point to the second. A curve is then constructed, starting at the current position and tangential to the first straight line. The curve is drawn such that it reaches the last point at a tangent to the second straight line. Example A shows the curve constructed, given current position A and the two points B and C.

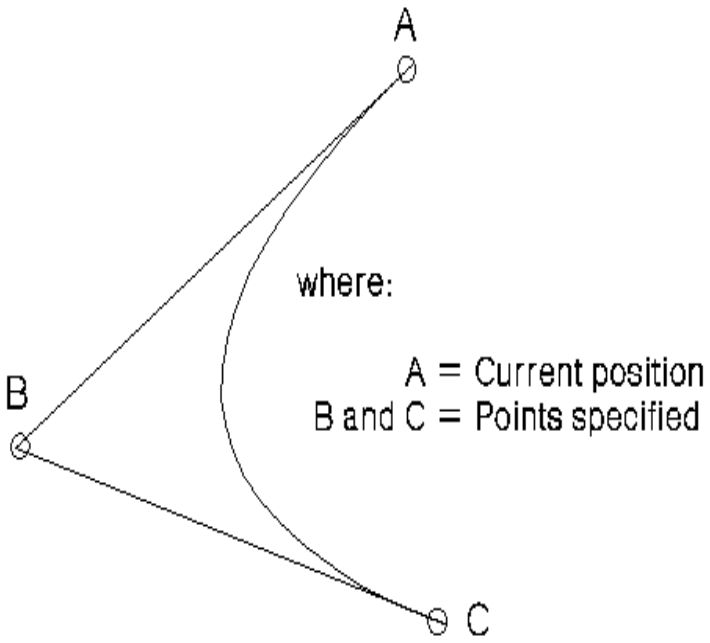
If more than two points are supplied, a series of imaginary straight lines is constructed through them (as in the GpiPolyLine function). All of the straight lines except the first and last are then divided in two at their mid-points. A series of curved fillets is then drawn, each starting at the end point of the last, at one of the mid-points. Example B shows the curve constructed, given current position A and three points B, C, and D.

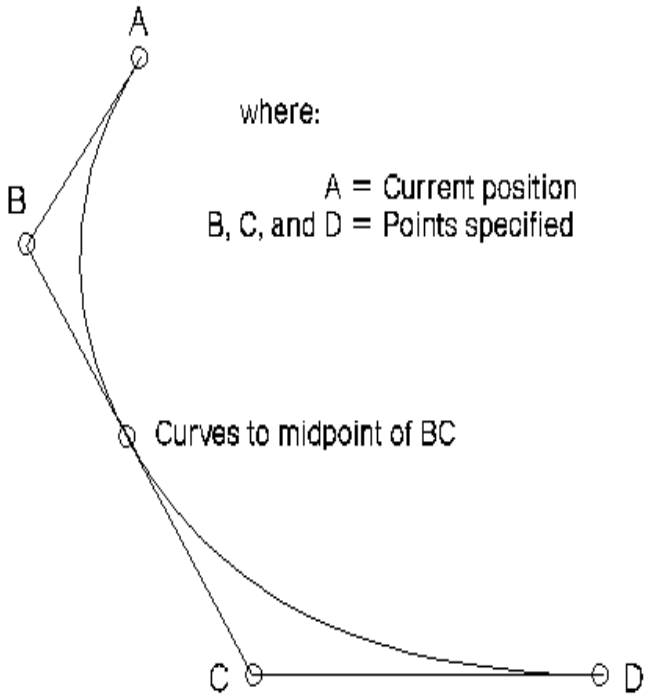
The current position is set to the last point.

Each individual fillet always lies within the area bounded by the start, end, and control points.

It is not an error for any of the points to be coincident.

The maximum number of fillets allowed in the polyfillet is more than 4 000.





GpiPolyFillet - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_NESTED_FIGURES (0x20A8)
Nested figures have been detected within a path definition.

GpiPolyFillet - Related Functions

Related Functions

- GpiPointArc
- GpiPolyFilletSharp
- GpiPolySpline
- GpiPop
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor

- GpiSetBackMix
- GpiSetColor
- GpiSetDefArcParams
- GpiSetDefAttrs
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix

GpiPolyFillet - Graphic Elements and Orders

Element Type: **OCODE_GCFLT**

Order: Fillet at Current Position
As many of these orders are generated as is necessary to hold the specified fillets.

GpiPolyFillet - Example Code

This example uses the GpiPolyFillet function to draw a curve with a loop. The four points are the four points of a rectangle. The curve is drawn from the lower-left corner, through the midpoint of the top edge, and back to the lower-right corner.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* start point                      */
POINTL aptl[3] = { 200, 100, 0, 100, 200, 0 }; /* curve points  */

GpiMove(hps, &ptlStart);        /* move to the lower-left corner */
GpiPolyFillet(hps, 3L, aptl);   /* draw the curve                */
```

GpiPolyFillet - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPolyFilletSharp

GpiPolyFilletSharp - Syntax

This function creates a fillet on a series of connected lines, with the first line starting at the current position. Subsequent points identify the end points of the lines.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lCount;      /*  Count of points. */
PPOINTL    aptlPoints;  /*  An array of points. */
PFIXED     afxPoints;   /*  Array of sharpness values. */
LONG       lHits;       /*  Correlation and error indicators. */

lHits = GpiPolyFilletSharp(hps, lCount, aptlPoints,
          afxPoints);
```

GpiPolyFilletSharp Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPolyFilletSharp Parameter - lCount

lCount (LONG) - input
Count of points.

This is the number of points specified in *aptlPoints* . It must be 2*f, where f is the number of fillets; the value must be a positive even number. Zero is valid but causes no output.

GpiPolyFilletSharp Parameter - aptlPoints

aptlPoints (PPOINTL) - input
An array of points.

These points are set as follows:

```
c1, e1, c2, e2, c3, e3, ... cf, ef
```

where:

cf	is the control point for the f'th fillet
ef	is the end point of the f'th fillet.

GpiPolyFilletSharp Parameter - afxPoints

afxPoints (PFIXED) - input
Array of sharpness values.

These give the sharpness of successive fillets.

GpiPolyFilletSharp Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

GpiPolyFilletSharp - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Count of points.

This is the number of points specified in *aptlPoints* . It must be 2*f, where f is the number of fillets; the value must be a positive even number. Zero is valid but causes no output.

aptlPoints (PPOINTL) - input
An array of points.

These points are set as follows:

```
c1, e1, c2, e2, c3, e3, ... cf, ef
```

where:

cf	is the control point for the f'th fillet
ef	is the end point of the f'th fillet.

afxPoints (PFIXED) - input
Array of sharpness values.

These give the sharpness of successive fillets.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK

	Successful
GPI_HITS	Correlate hits
GPI_ERROR	Error.

GpiPolyFilletSharp - Remarks

The first fillet is drawn using the two imaginary lines, one from current position to its control point (the first point specified in *aptlPoints*), and one from this point to the second point specified in *aptlPoints* . The fillet starts from current position, and ends at this second point. It is tangential to the first line at current position, and to the second line at the second point of *aptlPoints* . The sharpness of this fillet is given by the first element of the *afxPoints* array.

Each subsequent fillet is drawn starting from the end point of the previous fillet, and uses the next two lines in the sequence, in a similar way. Therefore two points and one sharpness value are required for each fillet.

The differences from GpiPolyFillet are:

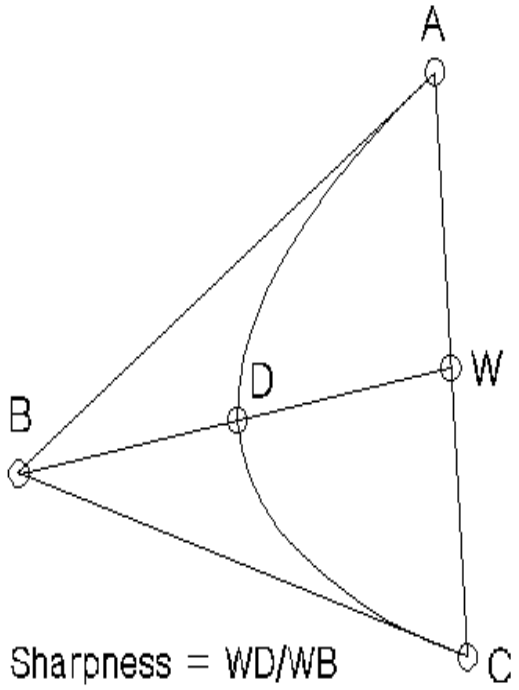
- The sharpness of each fillet is explicitly specified.
- Both the control and the end point of each fillet are explicitly specified.
- Adjacent fillets, generally, have a discontinuity in gradient, unless the points are chosen so that this is not the case.

The sharpness of each fillet is defined as follows. Let A and C be the start and end points, respectively, of the fillet, and let B be the control point. Let W be the mid-point of AC. Let D be the point where the fillet intersects WB.

```
sharpness = WD/DB
```

so that

> 1.0	means a hyperbola is drawn
= 1.0	means a parabola is drawn
< 1.0	means an ellipse is drawn.



On completion, the current position is the end point of the last line in the series. Each individual fillet always lies within the area bounded by the start, end, and control points.

It is not an error for any of the points to be coincident.

The maximum number of fillets allowed is more than 2 000.

GpiPolyFilletSharp - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_SHARPNESS_PARM (0x20CD)
An invalid sharpness parameter was specified with GpiPolyFilletSharp.

PMERR_INV_NESTED_FIGURES (0x20A8)
Nested figures have been detected within a path definition.

GpiPolyFilletSharp - Related Functions

Related Functions

- GpiPointArc
- GpiPolyFillet
- GpiPolySpline
- GpiPop
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefArcParams
- GpiSetDefAttrs
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix

GpiPolyFilletSharp - Graphic Elements and Orders

Element Type: **OCODE_GCSFLT**

Order: Sharp Fillet at Current Position

As many of these orders are generated as is necessary to hold the specified fillets.

GpiPolyFilletSharp - Example Code

This example uses the GpiPolyFilletSharp function to draw a curve with a loop. The curve is drawn within a rectangle. The sharpness values are chosen to draw the curve close to the control points.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* start of curve                   */
POINTL aptl[4]={ 100, 100, 200, 100, 0, 100, 200, 0};/* points  */
FIXED afx[2]={MAKEFIXED(4, 0), MAKEFIXED(4, 0)};/* sharpness    */

GpiMove(hps, &ptlStart);   /* move to first starting point         */
GpiPolyFilletSharp(hps,    /* presentation-space handle            */
    4L,                    /* 4 points in the array                */
    aptl,                  /* address of array of points           */
    afx);                  /* address of array of sharpness values */
```

GpiPolyFilletSharp - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPolygons

GpiPolygons - Syntax

This function draws a set of closed polygons.

```
#define INCL_GPIPOLYGON /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS         hps;        /*  Presentation-space handle. */
ULONG       ulCount;    /*  Number of polygons. */
```

```
PPOLYGON    paplgn;     /*  Array of polygons. */
ULONG       flOptions;  /*  Drawing options. */
ULONG       flModel;    /*  Drawing model. */
LONG        lHits;      /*  Correlation/error indicator. */

lHits = GpiPolygons(hps, ulCount, paplgn,
          flOptions, flModel);
```

GpiPolygons Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPolygons Parameter - ulCount

ulCount (ULONG) - input
Number of polygons.

Equal to the number of polygons in the polygons array. May be zero or positive, zero causes no output.

GpiPolygons Parameter - paplgn

paplgn (PPOLYGON) - input
Array of polygons.

An array of POLYGON structures.

GpiPolygons Parameter - flOptions

flOptions (ULONG) - input
Drawing options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can be ORed together to determine whether to draw boundary lines as well as the area interior.

Drawing boundary lines:

POLYGON_NOBOUNDARY	Does not draw boundary lines.
POLYGON_BOUNDARY	Draws boundary lines (the default).

Construction of the area interior:

POLYGON_ALTERNATE	Constructs interior in *alternate* mode (the default).

POLYGON_WINDING — Constructs interior in *winding* mode.

GpiPolygons Parameter - flModel

flModel (ULONG) - input
Drawing model.

POLYGON_INCL
: Fill is inclusive of bottom right (the default).

POLYGON_EXCL
: Fill is exclusive of bottom right.

This is provided to aid migration from other graphics models.

GpiPolygons Return Value - lHits

lHits (LONG) - returns
Correlation/error indicator.

GPI_OK
: Successful

GPI_HITS
: Correlate hits.

GPI_ERROR
: Error.

GpiPolygons - Parameters

hps (HPS) - input
Presentation-space handle.

ulCount (ULONG) - input
Number of polygons.

Equal to the number of polygons in the polygons array. May be zero or positive, zero causes no output.

paplgn (PPOLYGON) - input
Array of polygons.

An array of POLYGON structures.

flOptions (ULONG) - input
Drawing options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can be ORed together to determine whether to draw boundary lines as well as the area interior.

Drawing boundary lines:

POLYGON_NOBOUNDARY — Does not draw boundary lines.
POLYGON_BOUNDARY — Draws boundary lines (the default).

Construction of the area interior:

POLYGON_ALTERNATE	Constructs interior in *alternate* mode (the default).
POLYGON_WINDING	Constructs interior in *winding* mode.

flModel (ULONG) - input
Drawing model.

POLYGON_INCL
Fill is inclusive of bottom right (the default).
POLYGON_EXCL
Fill is exclusive of bottom right.

This is provided to aid migration from other graphics models.

lHits (LONG) - returns
Correlation/error indicator.

GPI_OK
Successful
GPI_HITS
Correlate hits.
GPI_ERROR
Error.

GpiPolygons - Remarks

The polygons are filled using the current AREABUNDLE structure values. For the first polygon, the current position is the first point. For all subsequent polygons all points which define the polygon are given explicitly. The polygons are automatically closed, if necessary, by drawing a line from the last vertex to the first.

The polygons may overlap, but that is not necessary.

GpiPolygons is not valid inside of an area.

GpiPolygons - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_AREA_CONTROL (0x2041)
An invalid options parameter was specified with GpiBeginArea.

PMERR_INV_IN_PATH (0x208B)
An attempt was made to issue a function invalid inside a path bracket.

PMERR_ALREADY_IN_AREA (0x2001)
An attempt was made to begin a new area while an existing area bracket was already open.

GpiPolygons - Graphic Elements and Orders

Element Type: **OCODE_GPOLYS**

Order: Polygons

GpiPolygons - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Graphic Elements and Orders
Glossary

GpiPolyLine

GpiPolyLine - Syntax

This function draws a series of straight lines starting at the current position and connecting the points specified.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lCount;      /*  Number of points. */
PPOINTL    aptlPoints;  /*  Array of points. */
LONG       lHits;       /*  Correlation and error indicators. */

lHits = GpiPolyLine(hps, lCount, aptlPoints);
```

GpiPolyLine Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPolyLine Parameter - lCount

lCount (LONG) - input
Number of points.

Must not be negative. Zero is valid but causes no output.

GpiPolyLine Parameter - aptlPoints

aptlPoints (PPOINTL) - input
Array of points.

GpiPolyLine Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPolyLine - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of points.

Must not be negative. Zero is valid but causes no output.

aptlPoints (PPOINTL) - input
Array of points.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPolyLine - Remarks

On completion, current position is set to the last point.

The maximum number of lines allowed in a polyline is device dependent, but is always greater than 3 500 for GPIF_LONG format spaces and always greater than 7 200 for GPIF_SHORT format spaces (see the PS_FORMAT of GpiCreatePS for the meaning of this format).

GpiPolyLine - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

PMERR_INV_NESTED_FIGURES (0x20A8)
: Nested figures have been detected within a path definition.

GpiPolyLine - Related Functions

Related Functions

- GpiBox
- GpiLine
- GpiMove
- GpiPolyLineDisjoint
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetCurrentPosition
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMix

GpiPolyLine - Graphic Elements and Orders

Element Type: **OCODE_GCLINE**

Note that GpiLine also generates this element type.

Order: Line at Current Position

As many of these orders are generated as is necessary to hold the specified points.

GpiPolyLine - Example Code

This example uses the GpiPolyLine function to draw a triangle.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                            /* presentation space handle*/
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 };
                                    /* vertices  */
GpiMove(hps, &ptlTriangle[2]);
                                    /* moves to end point (0, 0)*/
GpiPolyLine(hps, 3L, ptlTriangle); /* draws triangle   */
```

GpiPolyLine - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPolyLineDisjoint

GpiPolyLineDisjoint - Syntax

This function draws a series of disjoint straight lines using the end-point pairs specified.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lCount;      /*  Number of points. */
```

```
PPOINTL    aptlPoints;  /*  Array of points. */
LONG       lHits;       /*  Correlation/error indicator. */

lHits = GpiPolyLineDisjoint(hps, lCount, aptlPoints);
```

GpiPolyLineDisjoint Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPolyLineDisjoint Parameter - lCount

lCount (LONG) - input
Number of points.

Must be even and not negative. Zero is valid, but it causes no output. The maximum number of points allowed is system-dependent, but it is at least 7 000.

GpiPolyLineDisjoint Parameter - aptlPoints

aptlPoints (PPOINTL) - input
Array of points.

GpiPolyLineDisjoint Return Value - lHits

lHits (LONG) - returns
Correlation/error indicator.

GPI_OK
: Successful

GPI_HITS
: Correlate hit(s)

GPI_ERROR
: Error.

GpiPolyLineDisjoint - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of points.

Must be even and not negative. Zero is valid, but it causes no output. The maximum number of points allowed is system-dependent, but it is at least 7 000.

aptlPoints (PPOINTL) - input
Array of points.

lHits (LONG) - returns
Correlation/error indicator.

GPI_OK
Successful
GPI_HITS
Correlate hit(s)
GPI_ERROR
Error.

GpiPolyLineDisjoint - Remarks

The effect of this function is the same as the following sequence of calls:

```
GpiMove (hps, Points[0]);
GpiLine (hps, Points[1]);
GpiMove (hps, Points[2]);
GpiLine (hps, Points[3]);
   .
   .
   .
GpiMove (hps, Points[Count-2]);
GpiLine (hps, Points[Count-1]);
```

On completion, current position is set to the last point.

GpiPolyLineDisjoint - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_NESTED_FIGURES (0x20A8)
Nested figures have been detected within a path definition.

GpiPolyLineDisjoint - Related Functions

Related Functions

- GpiBox
- GpiLine
- GpiMove
- GpiPolyLine
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetCurrentPosition
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMix

GpiPolyLineDisjoint - Example Code

This example uses the GpiPolyLineDisjoint function to draw two lines.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions          */
#include <os2.h>

HPS hps;                /* presentation space handle                */
POINTL ptlLines[] = { 100, 100, 100, 200,      /* line 1     */
                      200, 100, 200, 200 };    /* line 2     */

GpiPolyLineDisjoint(hps, 4L, ptlLines);
                        /* draw lines */
```

GpiPolyLineDisjoint - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiPolyMarker

GpiPolyMarker - Syntax

This function draws markers with their centers at each of a series of specified positions.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lCount;      /*  Number of points. */
PPOINTL    aptlPoints;  /*  Array of points. */
LONG       lHits;       /*  Correlation and error indicators. */

lHits = GpiPolyMarker(hps, lCount, aptlPoints);
```

GpiPolyMarker Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPolyMarker Parameter - lCount

lCount (LONG) - input
Number of points.

Must not be negative. Zero is valid but causes no output.

GpiPolyMarker Parameter - aptlPoints

aptlPoints (PPOINTL) - input
Array of points.

A marker is drawn at each of these points.

GpiPolyMarker Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPolyMarker - Parameters

hps (HPS) - input
: Presentation-space handle.

lCount (LONG) - input
: Number of points.

Must not be negative. Zero is valid but causes no output.

aptlPoints (PPOINTL) - input
: Array of points.

A marker is drawn at each of these points.

lHits (LONG) - returns
: Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPolyMarker - Remarks

On completion, the current position is set to the position of the last marker in the series. The marker symbol is selected by the current values of the marker set and marker symbol attributes.

GpiPolyMarker - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

GpiPolyMarker - Related Functions

Related Functions

- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiMarker
- GpiSetMarker
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetMix

GpiPolyMarker - Graphic Elements and Orders

Element Type: **OCODE_GMRK**

Note that GpiMarker also generates this element type.

Order: Marker at Given Position

As many of these orders are generated as is necessary to hold the specified positions.

GpiPolyMarker - Example Code

This example uses the GpiPolyMarker function to draw a series of markers. It then uses the GpiPolyLine function to connect to markers with lines.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* start point                      */
POINTL aptl[5]={10, 8, 20, 17, 30, 28, 40, 51, 50, 46};/* points*/

GpiPolyMarker(hps, 5L, aptl);
GpiMove(hps, &ptlStart);
GpiPolyLine(hps, 5L, aptl);
```

GpiPolyMarker - Topics

Select an item:
Syntax
Parameters
Returns

Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPolySpline

GpiPolySpline - Syntax

This function creates a succession of Bezier splines.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lCount;      /*  Count of points. */
PPOINTL    aptlPoints;  /*  An array of points. */
LONG       lHits;       /*  Correlation and error indicators. */

lHits = GpiPolySpline(hps, lCount, aptlPoints);
```

GpiPolySpline Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPolySpline Parameter - lCount

lCount (LONG) - input
Count of points.

This is the number of points specified in *aptlPoints* . It must be three times the number of splines. The value must not be negative, and it must be divisible by 3. Zero is valid but causes no output.

GpiPolySpline Parameter - aptlPoints

aptlPoints (PPOINTL) - input
An array of points.

The points are given in this order:

```
c11, c12, e1, c21, c22, e2, ... cs1, cs2, es
```

where:

cs1	is the first control point of spline *s*
cs2	is the second control point of spline *s*
es	is the end point of spline *s*.

GpiPolySpline Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiPolySpline - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Count of points.

This is the number of points specified in *aptlPoints*. It must be three times the number of splines. The value must not be negative, and it must be divisible by 3. Zero is valid but causes no output.

aptlPoints (PPOINTL) - input
An array of points.

The points are given in this order:

```
c11, c12, e1, c21, c22, e2, ... cs1, cs2, es
```

where:

cs1	is the first control point of spline *s*
cs2	is the second control point of spline *s*
es	is the end point of spline *s*.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiPolySpline - Remarks

The first Bezier spline starts from the current position and goes to the third specified point, with the first and second points used as control points. Subsequent splines start from the ending point of the previous spline, and end at the next specified point but two, with the intervening points their first and second control points. It is the responsibility of the application to ensure that the gradient is continuous at each end and start point, if this is required.

On completion, the current position is set to the last specified point. Each individual spline always lies within the area bounded by the start, end, and control points.

It is not an error for any of the points to be coincident.

The maximum number of splines allowed is more than 2 500.

GpiPolySpline - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

PMERR_INV_NESTED_FIGURES (0x20A8)
: Nested figures have been detected within a path definition.

GpiPolySpline - Related Functions

Related Functions

- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPop
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefArcParams
- GpiSetDefAttrs
- GpiSetLineType

- GpiSetLineWidth
- GpiSetMix

GpiPolySpline - Graphic Elements and Orders

Element Type: **OCODE_GCBEZ**

Order: Bezier Curve at Given Position
As many of these orders are generated as is necessary to hold the specified splines.

GpiPolySpline - Example Code

This example uses the GpiPolySpline function to draw a curve. The curve is drawn within a skewed rectangle, with the bottom corners being the start and end points and the top corners being the control points.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* start point                      */
POINTL aptl[3] = { 0, 100, 200, 150, 200, 50 }; /* point array  */

GpiMove(hps, &ptlStart);         /* moves to start point       */
GpiPolySpline(hps,               /* presentation-space handle  */
    3L,                          /* 3 points in the array      */
    aptl);                       /* address of array of points */
```

GpiPolySpline - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPop

GpiPop - Syntax

This function restores the primitive attributes that have been saved on the stack.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lCount;  /*  Number of attributes to be restored. */
BOOL    rc;      /*  Success indicator. */

rc = GpiPop(hps, lCount);
```

GpiPop Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPop Parameter - lCount

lCount (LONG) - input
Number of attributes to be restored.

It must be greater or equal to 0.

GpiPop Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiPop - Parameters

hps (HPS) - input

Presentation-space handle.

lCount (LONG) - input
Number of attributes to be restored.

It must be greater or equal to 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiPop - Remarks

Each time a primitive attribute call (such as color, or line type) is issued and the attribute mode is set to AM_PRESERVE, the values are put into a "Last in, First out" stack.

This function can reset the current attribute values (starting with the last one set) to the previous value; this is known as "popping". This allows a called segment to change the values of the attributes, and allows them to be restored on return to the caller (an implicit GpiPop function is performed for each preserved attribute when returning from a called segment).

When inside an area or path definition, this function is only valid if the attribute being popped is valid inside an area or path definition.

Note: It is not possible to check whether the attribute to be popped is valid before issuing this function.

GpiPop - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_SEG_CALL_STACK_EMPTY (0x20FC)
A call stack empty condition was detected when attempting a pop function during GpiPop or segment drawing.

GpiPop - Related Functions

Related Functions

- GpiQueryAttrMode
- GpiQueryAttrs

- GpiQueryDefAttrs
- GpiRestorePS
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs

GpiPop - Graphic Elements and Orders

Element Type: **OCODE_GPOP**

Order: Pop

lCount of these orders are generated.

GpiPop - Example Code

This example uses the GpiPop function to restore the fill pattern and color attribute after painting a region.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#define INCL_GPIREGIONS         /* GPI region functions         */
#include <os2.h>

HPS hps;                /* presentation space handle            */
HRGN hrgn;              /* region handle                        */

/* preserves attributes on stack */
GpiSetAttrMode(hps, AM_PRESERVE);
    .
    .
    .
GpiSetColor(hps, CLR_RED);        /* sets color to red          */
GpiSetPattern(hps, PATSYM_DIAG1); /* sets pattern to a diagonal */
GpiPaintRegion(hps, hrgn);
GpiPop(hps, 2L);  /* restores values of last two attributes set */
```

GpiPop - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiPtInRegion

GpiPtInRegion - Syntax

This function checks whether a point lies within a region.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
HRGN       hrgn;       /*  Region handle. */
PPOINTL    pptlPoint;  /*  Point to be checked. */
LONG       lInside;    /*  Inside and error indicators. */

lInside = GpiPtInRegion(hps, hrgn, pptlPoint);
```

GpiPtInRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

GpiPtInRegion Parameter - hrgn

hrgn (HRGN) - input
Region handle.

GpiPtInRegion Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Point to be checked.

The point is in device coordinates.

GpiPtInRegion Return Value - lInside

llnside (LONG) - returns
Inside and error indicators.

PRGN_OUTSIDE
Not in region
PRGN_INSIDE
In region
PRGN_ERROR
Error.

GpiPtInRegion - Parameters

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) - input
Region handle.

pptlPoint (PPOINTL) - input
Point to be checked.

The point is in device coordinates.

llnside (LONG) - returns
Inside and error indicators.

PRGN_OUTSIDE
Not in region
PRGN_INSIDE
In region
PRGN_ERROR
Error.

GpiPtInRegion - Remarks

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiPtInRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiPtInRegion - Related Functions

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion
- GpiSetRegion

GpiPtInRegion - Example Code

This example uses GpiPtInRegion to determine if the point (150,150) lies within a region.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>

LONG   lInside;         /* inside/error indicator               */
HPS    hps;             /* Presentation-space handle            */
HRGN   hrgn;            /* handle for region                    */
POINTL pptlPoint = {150L,150L};/* point to be checked           */
RECTL arcl[3] = { 100, 100, 200, 200,        /* 1st rectangle   */
                  150, 150, 250, 250,        /* 2nd rectangle   */
                  200, 200, 300, 300 };      /* 3rd rectangle   */

/* create a region comprising three rectangles */
hrgn = GpiCreateRegion(hps, 3L, arcl);

lInside = GpiPtInRegion(hps, hrgn, &pptlPoint);
```

GpiPtInRegion - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code

Related Functions
Glossary

GpiPtVisible

GpiPtVisible - Syntax

This function checks whether a point is visible within the clipping region of the device associated with the specified presentation space.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;          /*  Presentation-space handle. */
PPOINTL    pptlPoint;    /*  Point to be checked. */
LONG       lVisibility;  /*  Visibility indicator. */

lVisibility = GpiPtVisible(hps, pptlPoint);
```

GpiPtVisible Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPtVisible Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Point to be checked.

The point is given in world coordinates.

GpiPtVisible Return Value - lVisibility

lVisibility (LONG) - returns
Visibility indicator.

PVIS_INVISIBLE

Not visible
PVIS_VISIBLE
Visible
PVIS_ERROR
Error.

GpiPtVisible - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - input
Point to be checked.

The point is given in world coordinates.

lVisibility (LONG) - returns
Visibility indicator.

PVIS_INVISIBLE
Not visible
PVIS_VISIBLE
Visible
PVIS_ERROR
Error.

GpiPtVisible - Remarks

For the purposes of this function, the clipping region is defined as the intersection between the application clipping region, and any other clipping, including windowing.

GpiPtVisible - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiPtVisible - Related Functions

Related Functions

- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiQueryPel
- GpiRectVisible
- GpiSetClipRegion
- GpiSetGraphicsField

GpiPtVisible - Example Code

This example uses GpiPtVisible to check whether (150,150) is visible within the clipping region of the device associated with the presentation space.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

LONG   lVisibility;     /* visibility indicator                 */
HPS    hps;             /* Presentation-space handle            */
POINTL pptlPoint = {150L,150L};/* point to be checked           */

lVisibility = GpiPtVisible(hps, &pptlPoint);
```

GpiPtVisible - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiPutData

GpiPutData - Syntax

This function passes a buffer of graphics orders to the current segment, or draws the orders, or both of these. For details of the orders, see Graphics Orders.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>
```

```
HPS      hps;      /*  Presentation-space handle. */
LONG     lFormat;  /*  Coordinate type used. */
PLONG    plCount;  /*  Length of graphic data. */
PBYTE    pbData;   /*  Orders to be copied. */
LONG     lHits;    /*  Correlation and error indicators. */

lHits = GpiPutData(hps, lFormat, plCount,
          pbData);
```

GpiPutData Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiPutData Parameter - lFormat

lFormat (LONG) - input
Coordinate type used.

This parameter can have one of the following values:

DFORM_NOCONV
No coordinate conversion performed
DFORM_S370SHORT
S/370 format short (2-byte) integers
DFORM_PCSHORT
PC format short (2-byte) integers
DFORM_PCLONG
PC format long (4-byte) integers.

GpiPutData Parameter - plCount

plCount (PLONG) - in/out
Length of graphic data.

Set by the application to the length of order data in *pbData*. If an incomplete order occurred, it is updated, on return, to the offset of the start of the incomplete order.

plCount must be greater or equal to 0 and less or equal to 64 152 bytes (63KB).

GpiPutData Parameter - pbData

pbData (PBYTE) - input
Orders to be copied.

GpiPutData Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiPutData - Parameters

hps (HPS) - input
Presentation-space handle.

lFormat (LONG) - input
Coordinate type used.

This parameter can have one of the following values:

DFORM_NOCONV
No coordinate conversion performed
DFORM_S370SHORT
S/370 format short (2-byte) integers
DFORM_PCSHORT
PC format short (2-byte) integers
DFORM_PCLONG
PC format long (4-byte) integers.

plCount (PLONG) - in/out
Length of graphic data.

Set by the application to the length of order data in *pbData* . If an incomplete order occurred, it is updated, on return, to the offset of the start of the incomplete order.

plCount must be greater or equal to 0 and less or equal to 64 152 bytes (63KB).

pbData (PBYTE) - input
Orders to be copied.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiPutData - Remarks

The orders passed may be added to the current segment, drawn immediately, or both, depending on the current drawing mode (see GpiSetDrawingMode), and whether the primitives are within a segment.

If there is an incomplete order at the end of the buffer, *plCount* is updated to point to the start of the incomplete order. The application can then concatenate this partial order in front of the next buffer.

The orders End Prolog and Set Viewing Transform are not allowed.

This function is valid within an element bracket (see GpiBeginElement). It can contain GpiBeginElement and GpiEndElement orders, while these are in the correct sequence with respect to the currently opened segment in segment store.

The data in the buffer is converted, if necessary, to the presentation space format (defined when the presentation space is first created; see GpiCreatePS).

This function is invalid if the editing mode (see GpiSetEditMode) is set to SEGEM_REPLACE, and also in SEGEM_INSERT mode if the element pointer is not pointing to the last element.

GpiPutData - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PUTDATA_FORMAT (0x20BB)
: An invalid format parameter was specified with GpiPutData.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_DATA_TOO_LONG (0x2016)
: An attempt was made to transfer more than the maximum permitted amount of data (64512 bytes) using GpiPutData, GpiGetData, or GpiElement.

PMERR_INV_ELEMENT_POINTER (0x206B)
: An attempt was made to issue GpiPutData with the element pointer not pointing at the last element.

PMERR_INV_REPLACE_MODE_FUNC (0x20C0)
: An attempt was made to issue GpiPutData with the editing mode set to SEGEM_REPLACE.

PMERR_ORDER_TOO_BIG (0x20E8)
: An internal size limit was exceeded while converting orders from short to long format during GpiPutData processing. An order was too long to convert.

GpiPutData - Related Functions

Related Functions

- GpiBeginElement
- GpiEndElement
- GpiGetData

GpiPutData - Example Code

This example uses the GpiPutData function to copy graphics orders from one segment to another.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

HPS hps;                /* presentation space handle            */
LONG fFormat = DFORM_NOCONV;/* do not convert coordinates       */
LONG offSegment = 0L;   /* offset in segment                    */
LONG offNextElement = 0;/* offset in segment to next element    */
LONG cb = 0L;           /* bytes retrieved                      */
BYTE abBuffer[512];     /* data buffer                          */


GpiOpenSegment(hps, 3L);   /* open segment to receive the data  */
do *lbrc.
    offSegment += cb;
    offNextElement = offSegment;
    cb = GpiGetData(hps, 2L, &offNextElement, fFormat, 512L, abBuffer);

    /* Put data in other segment. */

    if (cb > 0L) GpiPutData(hps, /* presentation-space handle    */
        fFormat,                 /* format of coordinates        */
        &cb,                     /* number of bytes in buffer    */
        abBuffer);            /* buffer with graphics-order data */

} while (cb > 0L);
GpiCloseSegment(hps);        /* close segment that received data */
```

GpiPutData - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryArcParams

GpiQueryArcParams - Syntax

This function returns the current arc parameters used to draw full, partial, and 3-point arcs.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS           hps;             /*  Presentation-space handle. */
PARCPARAMS    parcpArcParams;  /*  Arc parameters. */
BOOL          rc;              /*  Success indicator. */

rc = GpiQueryArcParams(hps, parcpArcParams);
```

GpiQueryArcParams Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryArcParams Parameter - parcpArcParams

parcpArcParams (PARCPARAMS) - output
Arc parameters.

GpiQueryArcParams Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryArcParams - Parameters

hps (HPS) - input
Presentation-space handle.

parcpArcParams (PARCPARAMS) - output
Arc parameters.

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryArcParams - Remarks

Arc parameters are set by GpiSetArcParams.

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryArcParams - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
: An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryArcParams - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetArcParams

GpiQueryArcParams - Example Code

This example uses GpiQueryArcParams to return the current arc parameters used to draw full, partial, and 3-point arcs. The example queries the arc parameters and assigns a variable to the P coefficient if the query succeeds.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions         */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                   */
HPS    hps;             /* Presentation-space handle           */
ARCPARAMS  parcpArcParams; /* Arc parameters                   */
```

```
LONG  lPcoefficient;    /* p coefficient of arc definition      */

fSuccess = GpiQueryArcParams(hps, &parcpArcParams);

/* if successful, assign value of P coefficient */
if (fSuccess == TRUE)
   lPcoefficient = parcpArcParams.lP;
```

GpiQueryArcParams - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryAttrMode

GpiQueryAttrMode - Syntax

This function returns the current value of the attribute mode, as set by GpiSetAttrMode.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lMode;  /*  Current attribute mode. */

lMode = GpiQueryAttrMode(hps);
```

GpiQueryAttrMode Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryAttrMode Return Value - lMode

lMode (LONG) - returns
Current attribute mode.

This parameter can have one of the following values:

>=0
Current attribute mode
AM_ERROR
Error.

GpiQueryAttrMode - Parameters

hps (HPS) - input
Presentation-space handle.

lMode (LONG) - returns
Current attribute mode.

This parameter can have one of the following values:

>=0
Current attribute mode
AM_ERROR
Error.

GpiQueryAttrMode - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryAttrMode - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetAttrMode

GpiQueryAttrMode - Example Code

This example uses GpiQueryAttrMode to return the current value of the attribute mode and sets a new mode using GpiSetAttrMode; after

the application has finished using the new mode, the original attribute mode is restored.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

LONG   lMode;           /* current attribute mode (or error)    */
HPS    hps;             /* Presentation-space handle            */

/* query current attribute mode */
lMode = GpiQueryAttrMode(hps);

/* set new mode */
GpiSetAttrMode(hps, AM_PRESERVE);
 .
 .
/* restore original mode */
GpiSetAttrMode(hps, lMode);
```

GpiQueryAttrMode - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Example Code
Related Functions
Glossary

GpiQueryAttrs

GpiQueryAttrs - Syntax

This function returns current attributes for the specified primitive type.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lPrimType;   /*  Primitive type. */
ULONG      flAttrMask;  /*  Attributes mask. */
PBUNDLE    ppbunAttrs;  /*  Attributes. */
LONG       lDefMask;    /*  Defaults mask. */

lDefMask = GpiQueryAttrs(hps, lPrimType, flAttrMask,
             ppbunAttrs);
```

GpiQueryAttrs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryAttrs Parameter - lPrimType

lPrimType (LONG) - input
Primitive type.

This is the type of primitive for which attributes are to be queried, as follows:

PRIM_LINE
Line and arc primitives
PRIM_CHAR
Character primitives
PRIM_MARKER
Marker primitives
PRIM_AREA
Area primitives
PRIM_IMAGE
Image primitives.

GpiQueryAttrs Parameter - flAttrMask

flAttrMask (ULONG) - input
Attributes mask.

Each flag that is set indicates that the corresponding flag in *lDefMask* is to be updated, and that if the corresponding attribute is not currently set to default, its value is to be returned in the *ppbunAttrs* buffer.

If all flags in *flAttrMask* are zero, the *ppbunAttrs* buffer address is not used.

GpiQueryAttrs Parameter - ppbunAttrs

ppbunAttrs (PBUNDLE) - output
Attributes.

ppbunAttrs is a buffer in which is returned the value of each non-default attribute for which the *flAttrMask* flag is set, in the order specified in GpiSetAttrs for the particular primitive type.

Only data for attributes for which the appropriate flag in *flAttrMask* is set is updated, so *ppbunAttrs* need only be large enough for the highest offset attribute to be returned (see GpiSetAttrs).

The data returned in *ppbunAttrs* for any attribute for which the *flAttrMask* flag is set, but which is currently set to default, is undefined.

GpiQueryAttrs Return Value - lDefMask

lDefMask (LONG) - returns
Defaults mask.

As *flDefMask* in GpiSetAttrs:

GPI_ALTERROR
Error occurred
Positive
Defaults mask, numeric value can be greater than or equal to 0.

GpiQueryAttrs - Parameters

hps (HPS) - input
Presentation-space handle.

lPrimType (LONG) - input
Primitive type.

This is the type of primitive for which attributes are to be queried, as follows:

PRIM_LINE
Line and arc primitives
PRIM_CHAR
Character primitives
PRIM_MARKER
Marker primitives
PRIM_AREA
Area primitives
PRIM_IMAGE
Image primitives.

flAttrMask (ULONG) - input
Attributes mask.

Each flag that is set indicates that the corresponding flag in *lDefMask* is to be updated, and that if the corresponding attribute is not currently set to default, its value is to be returned in the *ppbunAttrs* buffer.

If all flags in *flAttrMask* are zero, the *ppbunAttrs* buffer address is not used.

ppbunAttrs (PBUNDLE) - output
Attributes.

ppbunAttrs is a buffer in which is returned the value of each non-default attribute for which the *flAttrMask* flag is set, in the order specified in GpiSetAttrs for the particular primitive type.

Only data for attributes for which the appropriate flag in *flAttrMask* is set is updated, so *ppbunAttrs* need only be large enough for the highest offset attribute to be returned (see GpiSetAttrs).

The data returned in *ppbunAttrs* for any attribute for which the *flAttrMask* flag is set, but which is currently set to default, is undefined.

lDefMask (LONG) - returns
Defaults mask.

As *flDefMask* in GpiSetAttrs:

GPI_ALTERROR
Error occurred
Positive

Defaults mask, numeric value can be greater than or equal to 0.

GpiQueryAttrs - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**. This function returns a mask, similar in meaning to *flDefMask* in GpiSetAttrs. Each flag in the returned mask is updated if the corresponding flag in *flAttrMask* is set. It is set if the attribute is set to the default, otherwise it is reset. Other flags are undefined.

The parameters returned by this function may be used to reinstate exactly the same attributes as are queried, using GpiSetAttrs.

GpiQueryAttrs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PRIMITIVE_TYPE (0x20B9)
An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.

PMERR_UNSUPPORTED_ATTR (0x2109)
An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryAttrs - Related Functions

Related Functions

- GpiSetAttrs

GpiQueryAttrs - Example Code

This example uses the GpiQueryAttrs function to retrieve the current attributes for the line primitive.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
LINEBUNDLE lbnd;
```

```
LONG flDefMask;

flDefMask = GpiQueryAttrs(hps,  /* presentation-space handle */
    PRIM_LINE,                  /* line primitive            */
    LBB_COLOR |                 /* line color                */
    LBB_MIX_MODE |              /* color-mix mode            */
    LBB_WIDTH |                 /* line width                */
    LBB_GEOM_WIDTH |            /* geometric-line width      */
    LBB_TYPE |                  /* line style                */
    LBB_END |                   /* line-end style            */
    LBB_JOIN,                   /* line-join style           */
    &lbnd);                     /* buffer for attributes     */

if (flDefMask & LBB_COLOR)
   {
    /* The line color has the default value. */
   }
```

GpiQueryAttrs - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryBackColor

GpiQueryBackColor - Syntax

This function returns the current value of the (character) background color attribute, as set by the GpiSetBackColor function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lColor;  /*  Background color. */

lColor = GpiQueryBackColor(hps);
```

GpiQueryBackColor Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryBackColor Return Value - lColor

lColor (LONG) - returns
Background color.

If the presentation space is in RGB mode, *lColor* is an RGB value of background. If the presentation space is in index mode (as set through GpiCreateLogColorTable), *lColor* is a CLR_* index value.

CLR_ERROR
Error
CLR_DEFAULT
Default
Otherwise
Background color index.

GpiQueryBackColor - Parameters

hps (HPS) - input
Presentation-space handle.

lColor (LONG) - returns
Background color.

If the presentation space is in RGB mode, *lColor* is an RGB value of background. If the presentation space is in index mode (as set through GpiCreateLogColorTable), *lColor* is a CLR_* index value.

CLR_ERROR
Error
CLR_DEFAULT
Default
Otherwise
Background color index.

GpiQueryBackColor - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Valid CLR_* values are:

CLR_BACKGROUND (0L)
CLR_BLUE (1L)
CLR_RED (2L)
CLR_PINK (3L)
CLR_GREEN (4L)
CLR_CYAN (5L)
CLR_YELLOW (6L)
CLR_NEUTRAL (7L)
CLR_DARKGRAY (8L)
CLR_DARKBLUE (9L)

CLR_DARKRED (10L)
CLR_DARKPINK (11L)
CLR_DARKGREEN (12L)
CLR_DARKCYAN (13L)
CLR_BROWN (14L)
CLR_PALEGRAY (15L)

GpiQueryBackColor - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryBackColor - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetBackColor

GpiQueryBackColor - Example Code

This example uses GpiQueryBackColor to return the current value of the (character) background color attribute, as set by the GpiSetBackColor call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

LONG   lColor;          /* current background color (or error)  */
HPS    hps;             /* Presentation-space handle            */

lColor = GpiQueryBackColor(hps);
```

GpiQueryBackColor - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryBackMix

GpiQueryBackMix - Syntax

This function returns the current value of the (character) background color-mixing mode, as set by the GpiSetBackMix function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lMixMode;  /*  Background mix. */

lMixMode = GpiQueryBackMix(hps);
```

GpiQueryBackMix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryBackMix Return Value - lMixMode

lMixMode (LONG) - returns
Background mix.

BM_DEFAULT
Default
>0
Background mix mode
BM_ERROR
Error.

GpiQueryBackMix - Parameters

hps (HPS) - input
Presentation-space handle.

lMixMode (LONG) - returns
Background mix.

BM_DEFAULT
Default
>0
Background mix mode
BM_ERROR
Error.

GpiQueryBackMix - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryBackMix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryBackMix - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetBackMix

GpiQueryBackMix - Example Code

This example uses GpiQueryBackMix to return the current value of the (character) background color-mixing mode, as set by the GpiSetBackMix call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

LONG   lMixMode;        /* current background mix (or error)    */
HPS    hps;             /* Presentation-space handle            */

lMixMode = GpiQueryBackMix(hps);
```

GpiQueryBackMix - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryBitmapBits

GpiQueryBitmapBits - Syntax

This function transfers data from a bit map to application storage.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS             hps;             /*  Presentation-space handle. */
LONG            lScanStart;      /*  Starting line number. */
LONG            lScans;          /*  Number of scan lines to be returned. */
PBYTE           pbBuffer;        /*  Data area. */
PBITMAPINFO2    pbmiInfoTable;   /*  Bit-map information table. */
LONG            lScansReturned;  /*  Number of scan lines actually returned. */

lScansReturned = GpiQueryBitmapBits(hps, lScanStart,
                   lScans, pbBuffer, pbmiInfoTable);
```

GpiQueryBitmapBits Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryBitmapBits Parameter - lScanStart

lScanStart (LONG) - input
Starting line number.

Scan-line number at which the data transfer is to start, counting from zero as the bottom line. It must be greater or equal to 0.

GpiQueryBitmapBits Parameter - lScans

lScans (LONG) - input
Number of scan lines to be returned.

It must be greater or equal to 0.

GpiQueryBitmapBits Parameter - pbBuffer

pbBuffer (PBYTE) - output
Data area.

Data area into which the bit-map data is copied.

GpiQueryBitmapBits Parameter - pbmiInfoTable

pbmiInfoTable (PBITMAPINFO2) - in/out
Bit-map information table.

Storage must be provided for the associated color table.

GpiQueryBitmapBits Return Value - lScansReturned

lScansReturned (LONG) - returns

Number of scan lines actually returned.

>=0
: Number of scan lines actually returned

GPI_ALTERROR
: Error.

GpiQueryBitmapBits - Parameters

hps (HPS) - input
Presentation-space handle.

lScanStart (LONG) - input
Starting line number.

Scan-line number at which the data transfer is to start, counting from zero as the bottom line. It must be greater or equal to 0.

lScans (LONG) - input
Number of scan lines to be returned.

It must be greater or equal to 0.

pbBuffer (PBYTE) - output
Data area.

Data area into which the bit-map data is copied.

pbmiInfoTable (PBITMAPINFO2) - in/out
Bit-map information table.

Storage must be provided for the associated color table.

lScansReturned (LONG) - returns
Number of scan lines actually returned.

>=0
: Number of scan lines actually returned

GPI_ALTERROR
: Error.

GpiQueryBitmapBits - Remarks

The presentation space must be currently associated with a memory device context, which has a bit map currently selected.

The *pbmiInfoTable* must be initialized by the application with the values of *cbFix*, and also *cPlanes* and *cBitCount*, set to the format required. The standard bit-map formats are supported, plus any known to be supported by the device (see GpiQueryDeviceBitmapFormats). Each of the following fields must also be set by the application before issuing the call (unless the BITMAPINFO2 structure is truncated and the field is not present):

- *ulCompression*
- *usReserved*
- *usRecording*
- *usRendering*
- *ulColorEncoding*

This function returns the values of *cx*, *cy* (plus any other information, apart from that set by the application, for which space is available in the BITMAPINFO2 structure), and the color table array filled in by the system.

The bit-map data is converted where necessary.

pbBuffer must point to a storage area large enough to contain data for the requested number of scan lines. The amount of storage required for one scan line can be determined by calling GpiQueryBitmapParameters and using the returned values in the following formula:

```
((bitcount*bitmapwidth + 31)/32)*planes*4 bytes
```

The storage required for the entire bit map is this value multiplied by *cy*.

GpiQueryBitmapBits - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_INFO_TABLE (0x208F)
: An invalid bit-map info table was specified with a bit-map operation.

PMERR_NO_BITMAP_SELECTED (0x20E4)
: An attempt has been made to operate on a memory device context that has no bit map selected.

PMERR_INV_SCAN_START (0x20C4)
: An invalid scanstart parameter was specified with a bit-map function.

PMERR_INCORRECT_DC_TYPE (0x203C)
: An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

GpiQueryBitmapBits - Related Functions

Related Functions

- GpiSetBitmapBits

GpiQueryBitmapBits - Example Code

This example uses GpiQueryBitmapBits to copy the image data of a bit map from a presentation space associated with a memory device context.

```
#define INCL_GPIBITMAPS         /* GPI Bit-map functions         */
#define INCL_DOSMEMMGR          /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;                /* presentation space handle           */
BITMAPINFOHEADER2 bmp = { 16, 640, 350, 1, 1 }; /* info struct  */
ULONG cbBuffer, cbBitmapInfo; /* buffer lengths                */
PBYTE pbBuffer;         /* bit-map data buffer                 */
```

```
PBITMAPINFO2 pbmi;      /* info structure                        */

/*
 * Compute the size of the image-data buffer and the bit map
 * information structure.
 */
cbBuffer = (((bmp.cBitCount * bmp.cx) + 31) / 32)
    * 4 * bmp.cy * bmp.cPlanes;
cbBitmapInfo = sizeof(BITMAPINFO2) +
    (sizeof(RGB2) * (1 << bmp.cBitCount));

/*
 * Allocate memory for the image data-buffer and the bit map
 * information structure.
 */

DosAllocMem((VOID *)pbBuffer,cbBuffer,
            PAG_COMMIT | PAG_READ | PAG_WRITE);
DosAllocMem((VOID *)pbmi,cbBitmapInfo,
            PAG_COMMIT | PAG_READ | PAG_WRITE);

/* Copy the image data. */
pbmi->cbFix = 16L;
pbmi->cPlanes = 1;
pbmi->cBitCount = 1;
GpiQueryBitmapBits(hps, 0L, (LONG) bmp.cy, pbBuffer, pbmi);
```

GpiQueryBitmapBits - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryBitmapDimension

GpiQueryBitmapDimension - Syntax

This function returns the width and height of a bit map, as specified on a previous GpiSetBitmapDimension function.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HBITMAP    hbm;                  /*  Bit-map handle. */
PSIZEL     psizlBitmapDimension; /*  Size of bit map. */
BOOL       rc;                   /*  Success indicator. */

rc = GpiQueryBitmapDimension(hbm, psizlBitmapDimension);
```

GpiQueryBitmapDimension Parameter - hbm

hbm (HBITMAP) - input
Bit-map handle.

GpiQueryBitmapDimension Parameter - psizlBitmapDimension

psizlBitmapDimension (PSIZEL) - output
Size of bit map.

The width and height of the bit map in 0.1 millimeter units.

If not set by GpiSetBitmapDimension, zeros are returned.

GpiQueryBitmapDimension Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryBitmapDimension - Parameters

hbm (HBITMAP) - input
Bit-map handle.

psizlBitmapDimension (PSIZEL) - output
Size of bit map.

The width and height of the bit map in 0.1 millimeter units.

If not set by GpiSetBitmapDimension, zeros are returned.

rc (BOOL) - returns
Success indicator.

TRUE

Successful completion

FALSE

Error occurred.

GpiQueryBitmapDimension - Errors

Possible returns from WinGetLastError

PMERR_INV_HBITMAP (0x207B)
An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY (0x2032)
An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiQueryBitmapDimension - Related Functions

Related Functions

- GpiSetBitmapDimension

GpiQueryBitmapDimension - Example Code

This example uses GpiQueryBitmapDimension to return the width and height of a bit map, as specified on a previous GpiSetBitmapDimension call; if successful, it assigns the width to a variable.

```
#define INCL_GPIBITMAPS         /* Bit-map functions             */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                   */
HBITMAP hbm;            /* bit-map handle                      */
SIZEL  psizlBitmapDimension; /* size of bit map                */
LONG  lWidth;           /* width of bit map                    */

fSuccess = GpiQueryBitmapDimension(hbm, &psizlBitmapDimension);

/* if successful, assign value of bit-map width */
if (fSuccess == TRUE)
   lWidth = psizlBitmapDimension.cx;
```

GpiQueryBitmapDimension - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code

Related Functions
Glossary

GpiQueryBitmapHandle

GpiQueryBitmapHandle - Syntax

This function returns the handle of the bit map currently tagged with the specified local identifier (lcid).

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;    /*  Presentation-space handle. */
LONG       lLcid;  /*  Local identifier. */
HBITMAP    hbm;    /*  Bit-map handle. */

hbm = GpiQueryBitmapHandle(hps, lLcid);
```

GpiQueryBitmapHandle Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryBitmapHandle Parameter - lLcid

lLcid (LONG) - input
Local identifier.

It must be greater than 0 and less than 255.

GpiQueryBitmapHandle Return Value - hbm

hbm (HBITMAP) - returns
Bit-map handle.

<>0

Bit-map handle

GPI_ERROR
Error.

GpiQueryBitmapHandle - Parameters

hps (HPS) - input
Presentation-space handle.

lLcid (LONG) - input
Local identifier.

It must be greater than 0 and less than 255.

hbm (HBITMAP) - returns
Bit-map handle.

<>0
Bit-map handle

GPI_ERROR
Error.

GpiQueryBitmapHandle - Remarks

An error is raised if a bit map is not currently tagged with the specified *lLcid*.

GpiQueryBitmapHandle - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SETID (0x20CA)
An invalid setid parameter was specified.

PMERR_ID_HAS_NO_BITMAP (0x2036)
No bit map was tagged with the setid specified on a GpiQueryBitmapHandle function.

GpiQueryBitmapHandle - Related Functions

Related Functions

- GpiSetBitmapId

GpiQueryBitmapHandle - Example Code

This example uses GpiQueryBitmapHandle to return the handle of the bit map currently tagged with the specified local identifier (lcid) set by GpiSetBitmapId.

```
#define INCL_GPIBITMAPS         /* Bit-map functions            */
#include <os2.h>

HBITMAP hbm;            /* bit-map handle                       */
HPS     hps;            /* presentation-space handle            */
LONG    lLcid;          /* local identifier                     */

hbm = GpiQueryBitmapHandle(hps, lLcid);
```

GpiQueryBitmapHandle - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryBitmapInfoHeader

GpiQueryBitmapInfoHeader - Syntax

This function returns information about a bit map identified by the bit-map handle.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HBITMAP               hbm;       /*  Bit-map handle. */
PBITMAPINFOHEADER2    pbmpData;  /*  Bit-map information header. */
BOOL                  rc;        /*  Success indicator. */

rc = GpiQueryBitmapInfoHeader(hbm, pbmpData);
```

GpiQueryBitmapInfoHeader Parameter - hbm

hbm (HBITMAP) - input
Bit-map handle.

GpiQueryBitmapInfoHeader Parameter - pbmpData

pbmpData (PBITMAPINFOHEADER2) - in/out
Bit-map information header.

This is a structure, that on return, is filled with data for the specified bit map.

GpiQueryBitmapInfoHeader Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryBitmapInfoHeader - Parameters

hbm (HBITMAP) - input
Bit-map handle.

pbmpData (PBITMAPINFOHEADER2) - in/out
Bit-map information header.

This is a structure, that on return, is filled with data for the specified bit map.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryBitmapInfoHeader - Remarks

The *length* field of the BITMAPINFOHEADER2 structure must be set by the application before performing this function.

Note: This function should be used in preference to the GpiQueryBitmapParameters function.

GpiQueryBitmapInfoHeader - Errors

Possible returns from WinGetLastError

PMERR_INV_HBITMAP (0x207B)
An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY (0x2032)
An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiQueryBitmapInfoHeader - Example Code

This example uses GpiQueryBitmapInfoHeader to return information about a bit map identified by the bit-map handle; if successful, it uses this information to create a new bit map via GpiCreateBitmap.

```
#define INCL_GPIBITMAPS         /* Bit-map functions             */
#include <os2.h>

HPS     hps;            /* presentation-space handle           */
BOOL    fSuccess;       /* success indicator                   */
HBITMAP hbm;            /* bit-map handle                      */
HBITMAP hbmNew;         /* bit-map handle                      */
BITMAPINFOHEADER2  pbmp2Data; /* Bit-map information header     */
PBYTE pb;               /* address of bit-map image data in
                           resource                           */

/* set size of info structure */
pbmp2Data.cbFix = 16L;

fSuccess = GpiQueryBitmapInfoHeader(hbm, &pbmp2Data);

/* use information to create bit map */
hbmNew = GpiCreateBitmap(hps,    /* presentation space         */
                  &pbmp2Data,    /* bit-map information header */
                  CBM_INIT,      /* initialize the bit map     */
                  pb,            /* bit-map data               */
                  (PBITMAPINFO2)&pbmp2Data);
                                 /* bit-map information table  */
```

GpiQueryBitmapInfoHeader - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryBitmapParameters

GpiQueryBitmapParameters - Syntax

This function returns information about a bit map identified by the bit-map handle.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HBITMAP             hbm;       /*  Bit-map handle. */
PBITMAPINFOHEADER   pbmpData;  /*  Bit-map information header. */
BOOL                rc;        /*  Success indicator. */

rc = GpiQueryBitmapParameters(hbm, pbmpData);
```

GpiQueryBitmapParameters Parameter - hbm

hbm (HBITMAP) - input
Bit-map handle.

GpiQueryBitmapParameters Parameter - pbmpData

pbmpData (PBITMAPINFOHEADER) - in/out
Bit-map information header.

This is a structure, that on return, is filled with data for the specified bit map. The structure includes the elements (width, height, planes, bitcount) of a bit-map information table.

GpiQueryBitmapParameters Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryBitmapParameters - Parameters

hbm (HBITMAP) - input
Bit-map handle.

pbmpData (PBITMAPINFOHEADER) - in/out
Bit-map information header.

This is a structure, that on return, is filled with data for the specified bit map. The structure includes the elements (width, height, planes, bitcount) of a bit-map information table.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryBitmapParameters - Remarks

The *cbFix* field of the BITMAPINFOHEADER structure must be set by the application before performing this function.

GpiQueryBitmapParameters - Errors

Possible returns from WinGetLastError

PMERR_INV_HBITMAP (0x207B)
An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY (0x2032)
An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiQueryBitmapParameters - Related Functions

Related Functions

- GpiCreateBitmap

GpiQueryBitmapParameters - Example Code

This example uses GpiQueryBitmapParameters to return information about a bit map identified by the bit-map handle; if successful, it assigns the width field to a variable.

```
#define INCL_GPIBITMAPS         /* Bit-map functions             */
#include <os2.h>

BOOL    fSuccess;       /* success indicator                    */
HBITMAP hbm;            /* bit-map handle                       */
BITMAPINFOHEADER  pbmpData; /* bit-map information header     */
USHORT usWidth;         /* width of bit map                     */

/* set size of info structure */
pbmpData.cbFix = sizeof(BITMAPINFOHEADER);

fSuccess = GpiQueryBitmapParameters(hbm, &pbmpData);

/* if successful, assign value of bit-map width */
if (fSuccess == TRUE)
   usWidth = pbmpData.cx;
```

GpiQueryBitmapParameters - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryBoundaryData

GpiQueryBoundaryData - Syntax

This function returns the boundary data.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;           /*  Presentation-space handle. */
PRECTL    prclBoundary;  /*  Pointer to a rectangle structure in which the boundary data is returned. */
BOOL      rc;            /*  Success indicator. */

rc = GpiQueryBoundaryData(hps, prclBoundary);
```

GpiQueryBoundaryData Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryBoundaryData Parameter - prclBoundary

prclBoundary (PRECTL) - output
Pointer to a rectangle structure in which the boundary data is returned.

GpiQueryBoundaryData Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryBoundaryData - Parameters

hps (HPS) - input
Presentation-space handle.

prclBoundary (PRECTL) - output
Pointer to a rectangle structure in which the boundary data is returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryBoundaryData - Remarks

This function returns the boundary data set upon completion of the last boundary calculation. Boundary data is returned as the coordinates in model space.

Boundary data is inclusive. A null boundary is indicated if *xLeft* is greater than *xRight*, or if *yBottom* is greater than *yTop*. After GpiResetBoundaryData, *xLeft* and *yBottom* are the maximum positive numbers, and *xRight* and *yTop* are the maximum negative numbers.

GpiQueryBoundaryData - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_COORDINATE_OVERFLOW (0x2014)
: An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryBoundaryData - Related Functions

Related Functions

- GpiResetBoundaryData
- GpiSetDrawControl

GpiQueryBoundaryData - Example Code

This example uses the GpiQueryBoundaryData function to retrieve the rectangle enclosing the output. The boundary data is then used to draw a border around the output.

```
#define INCL_GPICORRELATION
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#define INCL_GPICONTROL         /* GPI control Functions        */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 0, 0 }; /* first vertex                     */
POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 }; /* vertices  */
RECTL rcl;              /* rectangle                            */

GpiSetDrawControl(hps,
    DCTL_BOUNDARY, DCTL_ON);      /* accumulate boundary data   */

GpiMove(hps, &ptlStart);          /* produce output             */
GpiPolyLine(hps, 3L, ptlTriangle);

GpiQueryBoundaryData(hps, &rcl);  /* copy boundary data to rcl  */
if (rcl.xLeft < rcl.xRight) {     /* verify output exists*/
    ptlStart.x = rcl.xLeft; ptlStart.y = rcl.yBottom;
    GpiMove(hps, &ptlStart);      /* move to lower-right corner */
    ptlStart.x = rcl.xRight; ptlStart.y = rcl.yTop;
    GpiBox(hps, DRO_OUTLINE, &ptlStart, 0L, 0L); /* draw border */
```

```
}
```

GpiQueryBoundaryData - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryCharAngle

GpiQueryCharAngle - Syntax

This function returns the current value of the character baseline angle.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS           hps;          /*  Presentation-space handle. */
PGRADIENTL    pgradlAngle;  /*  Baseline angle. */
BOOL          rc;           /*  Success indicator. */

rc = GpiQueryCharAngle(hps, pgradlAngle);
```

GpiQueryCharAngle Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharAngle Parameter - pgradlAngle

pgradlAngle (PGRADIENTL) - output
Baseline angle.

A point, relative to (0,0), that defines the character baseline angle vector.

If the character angle is currently set to the default value, (0,0) is returned.

GpiQueryCharAngle Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharAngle - Parameters

hps (HPS) - input
Presentation-space handle.

pgradlAngle (PGRADIENTL) - output
Baseline angle.

A point, relative to (0,0), that defines the character baseline angle vector.

If the character angle is currently set to the default value, (0,0) is returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharAngle - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharAngle - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCharAngle - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetCharAngle

GpiQueryCharAngle - Example Code

This example uses GpiQueryCharAngle to return the current value of the character baseline angle; if successful, it places the x-component in a variable.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
LONG   lxComponent;     /* x component of baseline angle        */
GRADIENTL  pgradlAngle; /* Baseline angle                       */

fSuccess = GpiQueryCharAngle(hps, &pgradlAngle);

if (fSuccess == TRUE)
   lxComponent = pgradlAngle.x;
```

GpiQueryCharAngle - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryCharBox

GpiQueryCharBox - Syntax

This function returns the current value of the character box attribute, as set by the GpiSetCharBox function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;         /*  Presentation-space handle. */
PSIZEF    psizfxSize;  /*  Character-box size. */
BOOL      rc;          /*  Success indicator. */

rc = GpiQueryCharBox(hps, psizfxSize);
```

GpiQueryCharBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharBox Parameter - psizfxSize

psizfxSize (PSIZEF) - output
Character-box size.

If the character box is currently set to the default, the default size is returned. This is the size returned by DevQueryCaps (CAPS_GRAPHICS_CHAR_WIDTH and CAPS_GRAPHICS_CHAR_HEIGHT), converted to presentation page space.

GpiQueryCharBox Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharBox - Parameters

hps (HPS) - input
Presentation-space handle.

psizfxSize (PSIZEF) - output
Character-box size.

If the character box is currently set to the default, the default size is returned. This is the size returned by DevQueryCaps (CAPS_GRAPHICS_CHAR_WIDTH and CAPS_GRAPHICS_CHAR_HEIGHT), converted to presentation page space.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharBox - Remarks

In general this function does not return the same box as GpiQueryTextBox for an average-sized character. For outline fonts the character-box attribute is mapped to a particular font dimension related to the point size, for raster fonts it does not correspond to any font metric. (See GpiSetCharMode).

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCharBox - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetCharBox

GpiQueryCharBox - Example Code

This example uses GpiQueryCharBox to return the current value of the character box attribute, as set by the GpiSetCharBox call; if successful, places the box width in a variable.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
SIZEF  psizfxSize;      /* Character-box size                   */
FIXED  lWidth;          /* character box width                  */

fSuccess = GpiQueryCharBox(hps, &psizfxSize);

if (fSuccess == TRUE)
   lWidth = psizfxSize.cx;
```

GpiQueryCharBox - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryCharBreakExtra

GpiQueryCharBreakExtra - Syntax

This function returns the current value of the character-break-extra attribute, as set by the GpiSetCharBreakExtra function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;         /*  Presentation-space handle. */
PFIXED    BreakExtra;  /*  Character-break-extra attribute value. */
BOOL      rc;          /*  Success indicator. */

rc = GpiQueryCharBreakExtra(hps, BreakExtra);
```

GpiQueryCharBreakExtra Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharBreakExtra Parameter - BreakExtra

BreakExtra (PFIXED) - output
Character-break-extra attribute value.

GpiQueryCharBreakExtra Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharBreakExtra - Parameters

hps (HPS) - input
Presentation-space handle.

BreakExtra (PFIXED) - output
Character-break-extra attribute value.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharBreakExtra - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharBreakExtra - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

GpiQueryCharBreakExtra - Example Code

This example uses GpiQueryCharBreakExtra to return the current value of the character-break-extra attribute, as set by the GpiSetCharBreakExtra call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
FIXED  pfxBreakExtra;   /* Character-break-extra attribute value*/

fSuccess = GpiQueryCharBreakExtra(hps, &pfxBreakExtra);
```

GpiQueryCharBreakExtra - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryCharDirection

GpiQueryCharDirection - Syntax

This call returns the current value of the character direction attribute, as set by the GpiSetCharDirection function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;         /*  Presentation-space handle. */
LONG    lDirection;  /*  Character direction. */

lDirection = GpiQueryCharDirection(hps);
```

GpiQueryCharDirection Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharDirection Return Value - lDirection

lDirection (LONG) - returns
Character direction.

This parameter can have one of the following values:

CHDIRN_DEFAULT
Default
>0
Character direction
CHDIRN_ERROR
Error.

GpiQueryCharDirection - Parameters

hps (HPS) - input
Presentation-space handle.

lDirection (LONG) - returns
Character direction.

This parameter can have one of the following values:

CHDIRN_DEFAULT
Default
>0
Character direction
CHDIRN_ERROR
Error.

GpiQueryCharDirection - Remarks

This call is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharDirection - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
: An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCharDirection - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetCharDirection

GpiQueryCharDirection - Example Code

This example uses GpiQueryCharDirection to return the current value of the character direction attribute, as set by the GpiSetCharDirection call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

LONG  lDirection;       /* character direction (or error)       */
HPS    hps;             /* Presentation-space handle            */

lDirection = GpiQueryCharDirection(hps);
```

GpiQueryCharDirection - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryCharExtra

GpiQueryCharExtra - Syntax

This function returns the current value of the character-extra attribute, as set by the GpiSetCharExtra function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;    /*  Presentation-space handle. */
PFIXED    Extra;  /*  Character-extra attribute value. */
BOOL      rc;     /*  Success indicator. */

rc = GpiQueryCharExtra(hps, Extra);
```

GpiQueryCharExtra Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharExtra Parameter - Extra

Extra (PFIXED) - output
Character-extra attribute value.

GpiQueryCharExtra Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharExtra - Parameters

hps (HPS) - input
Presentation-space handle.

Extra (PFIXED) - output
Character-extra attribute value.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharExtra - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharExtra - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

GpiQueryCharExtra - Example Code

This example uses GpiQueryCharExtra to return the current value of the character-extra attribute, as set by the GpiSetCharExtra call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
FIXED  pfxExtra;        /* Character-extra attribute value      */

fSuccess = GpiQueryCharExtra(hps, &pfxExtra);
```

GpiQueryCharExtra - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryCharMode

GpiQueryCharMode - Syntax

This function returns the current value of the character-mode attribute, as set by the GpiSetCharMode function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lMode;  /*  Character mode. */

lMode = GpiQueryCharMode(hps);
```

GpiQueryCharMode Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharMode Return Value - lMode

lMode (LONG) - returns
Character mode.

CM_DEFAULT
Default
>0
Character mode
CM_ERROR
Error.

GpiQueryCharMode - Parameters

hps (HPS) - input
Presentation-space handle.

lMode (LONG) - returns
Character mode.

CM_DEFAULT
Default
>0
Character mode
CM_ERROR
Error.

GpiQueryCharMode - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharMode - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCharMode - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetCharMode

GpiQueryCharMode - Example Code

This example uses GpiQueryCharMode to return the current value of the character mode attribute, as set by the GpiSetCharMode call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

LONG   lMode;           /* character mode attribute             */
HPS    hps;             /* Presentation-space handle            */

lMode = GpiQueryCharMode(hps);
```

GpiQueryCharMode - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryCharSet

GpiQueryCharSet - Syntax

This function returns the character-set local identifier (lcid), as set by the GpiSetCharSet function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lLcid;  /*  Character-set local identifier. */

lLcid = GpiQueryCharSet(hps);
```

GpiQueryCharSet Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharSet Return Value - lLcid

lLcid (LONG) - returns
Character-set local identifier.

LCID_DEFAULT
Default
>0
Local identifier
LCID_ERROR
Error.

GpiQueryCharSet - Parameters

hps (HPS) - input
Presentation-space handle.

lLcid (LONG) - returns
Character-set local identifier.

LCID_DEFAULT
Default
>0
Local identifier
LCID_ERROR
Error.

GpiQueryCharSet - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharSet - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCharSet - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetCharSet

GpiQueryCharSet - Example Code

This example uses GpiQueryCharSet to return the character-set local identifier (lcid), as set by the GpiSetCharSet call.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

LONG  lLcid;            /* character set lcid (or error)        */
HPS    hps;             /* Presentation-space handle            */

lLcid = GpiQueryCharSet(hps);
```

GpiQueryCharSet - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryCharShear

GpiQueryCharShear - Syntax

This function returns the value of the current character-shear angle, as set by the GpiSetCharShear function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlShear;  /*  Character shear. */
BOOL       rc;         /*  Success indicator. */

rc = GpiQueryCharShear(hps, pptlShear);
```

GpiQueryCharShear Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharShear Parameter - pptlShear

pptlShear (PPOINTL) - output
Character shear.

A point, relative to (0,0), that defines the character shear vector.

If the character shear is currently set to the default, (0,1) is returned.

GpiQueryCharShear Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharShear - Parameters

hps (HPS) - input
Presentation-space handle.

pptlShear (PPOINTL) - output
Character shear.

A point, relative to (0,0), that defines the character shear vector.

If the character shear is currently set to the default, (0,1) is returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharShear - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharShear - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCharShear - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetCharShear

GpiQueryCharShear - Example Code

This example uses GpiQueryCharShear to return the value of the current character-shear angle, as set by the GpiSetCharShear call; if successful, it assigns the x-coordinate of the returned vector to a variable.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
POINTL pptlShear;       /* character shear                      */
LONG   lxCoord;         /* shear angle vector x coordinate      */

fSuccess = GpiQueryCharShear(hps, &pptlShear);

if (fSuccess == TRUE)
   lxCoord = pptlShear.x;
```

GpiQueryCharShear - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryCharStringPos

GpiQueryCharStringPos - Syntax

This function processes a string as if it is being drawn under the current character attributes using GpiCharStringPos, and returns the positions in the string at which each character would be drawn.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;            /*  Presentation-space handle. */
ULONG      flOptions;      /*  Option flag. */
LONG       lCount;         /*  Length of the string. */
PCH        pchString;      /*  Character string to be examined. */
PLONG      alXincrements;  /*  Vector of x-increment values. */
PPOINTL    aptlPositions;  /*  Array of points. */
BOOL       rc;             /*  Success indicator. */

rc = GpiQueryCharStringPos(hps, flOptions,
       lCount, pchString, alXincrements, aptlPositions);
```

GpiQueryCharStringPos Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharStringPos Parameter - flOptions

flOptions (ULONG) - input
Option flag.

CHS_VECTOR
Increments vector supplied (*alXincrements*). If 0, *alXincrements* is ignored.

GpiQueryCharStringPos Parameter - lCount

lCount (LONG) - input
Length of the string.

It must be greater or equal to 0 and less or equal to 512.

GpiQueryCharStringPos Parameter - pchString

pchString (PCH) - input
Character string to be examined.

GpiQueryCharStringPos Parameter - alXincrements

alXincrements (PLONG) - input
Vector of x-increment values.

These are signed values in world coordinates. Any negative values are treated as if they were 0. This parameter is ignored if CHS_VECTOR is not set.

GpiQueryCharStringPos Parameter - aptlPositions

aptlPositions (PPOINTL) - output
Array of points.

The positions of each character in world coordinates. The first point returned is the initial current position, and the last point is the new current position if the string has been drawn.

GpiQueryCharStringPos Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharStringPos - Parameters

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Option flag.

CHS_VECTOR
Increments vector supplied (*alXincrements*). If 0, *alXincrements* is ignored.

lCount (LONG) - input
Length of the string.

It must be greater or equal to 0 and less or equal to 512.

pchString (PCH) - input
Character string to be examined.

alXincrements (PLONG) - input
Vector of x-increment values.

These are signed values in world coordinates. Any negative values are treated as if they were 0. This parameter is ignored if CHS_VECTOR is not set.

aptlPositions (PPOINTL) - output
Array of points.

The positions of each character in world coordinates. The first point returned is the initial current position, and the last point is the new current position if the string has been drawn.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharStringPos - Remarks

A vector of increments can be specified, allowing control over the positioning of each character after the first. These are distances measured in world coordinates (along the baseline for left-to-right and right-to-left character directions, and along the shearline for top-to-bottom and bottom-to-top). The i'th increment is the distance of the reference point of the (i+1)'th character from the reference point of the i'th. The last increment may be needed to update current position.

These increments, if specified, set the widths of each character.

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharStringPos - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_CHAR_POS_OPTIONS (0x204E)
An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_COORDINATE_OVERFLOW (0x2014)
An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCharStringPos - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPosAt
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet

- GpiSetCharShear

GpiQueryCharStringPos - Example Code

This example calls the GpiQueryCharStringPos function to determine the location of each character in the string. Vector increments are not used.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
CHAR szString[] = "Sample string";
POINTL aptl[sizeof(szString) + 1];

GpiQueryCharStringPos(hps,     /* presentation-space handle      */
    0L,                        /* does not use vector increments */
    sizeof(szString),          /* number of characters in string */
    szString,                  /* character string               */
    NULL,                      /* no vector increments           */
    aptl);                     /* array of structures for points */
```

GpiQueryCharStringPos - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiQueryCharStringPosAt

GpiQueryCharStringPosAt - Syntax

This function processes a string as if it is being drawn under the current character attributes using GpiCharStringPosAt, and returns the positions in the string at which each character would be drawn.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;            /*  Presentation-space handle. */
PPOINTL    pptlStart;      /*  Starting position. */
ULONG      flOptions;      /*  Option flags. */
```

```
LONG       lCount;         /*  Length of the string. */
PCH        pchString;      /*  Character string to be examined. */
PLONG      alXincrements;  /*  Array of x increment values. */
PPOINTL    aptlPositions;  /*  Array of points. */
BOOL       rc;             /*  Success indicator. */

rc = GpiQueryCharStringPosAt(hps, pptlStart,
       flOptions, lCount, pchString, alXincrements,
       aptlPositions);
```

GpiQueryCharStringPosAt Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCharStringPosAt Parameter - pptlStart

pptlStart (PPOINTL) - input
Starting position.

GpiQueryCharStringPosAt Parameter - flOptions

flOptions (ULONG) - input
Option flags.

CHS_VECTOR
Increments vector supplied (*alXincrements*). If 0, *alXincrements* is ignored.

GpiQueryCharStringPosAt Parameter - lCount

lCount (LONG) - input
Length of the string.

It must be greater or equal to 0 and less or equal to 512.

GpiQueryCharStringPosAt Parameter - pchString

pchString (PCH) - input
Character string to be examined.

GpiQueryCharStringPosAt Parameter - alXincrements

alXincrements (PLONG) - input
Array of x increment values.

These are signed values in world coordinates. Any negative values are treated as if they were 0. This parameter is ignored if CHS_VECTOR is not set.

GpiQueryCharStringPosAt Parameter - aptlPositions

aptlPositions (PPOINTL) - output
Array of points.

Array of points, in which the positions of each character in world coordinates are returned. The first point returned is the initial current position, and the last point is the new current position if the string has been drawn.

GpiQueryCharStringPosAt Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharStringPosAt - Parameters

hps (HPS) - input
Presentation-space handle.

pptlStart (PPOINTL) - input
Starting position.

flOptions (ULONG) - input
Option flags.

CHS_VECTOR
Increments vector supplied (*alXincrements*). If 0, *alXincrements* is ignored.

lCount (LONG) - input

Length of the string.

It must be greater or equal to 0 and less or equal to 512.

pchString (PCH) - input
Character string to be examined.

alXincrements (PLONG) - input
Array of x increment values.

These are signed values in world coordinates. Any negative values are treated as if they were 0. This parameter is ignored if CHS_VECTOR is not set.

aptlPositions (PPOINTL) - output
Array of points.

Array of points, in which the positions of each character in world coordinates are returned. The first point returned is the initial current position, and the last point is the new current position if the string has been drawn.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCharStringPosAt - Remarks

A vector of increments can be specified, allowing control over the positioning of each character after the first. These are distances measured in world coordinates (along the baseline for left-to-right and right-to-left character directions, and along the shearline for top-to-bottom and bottom-to-top). The i'th increment is the distance of the reference point of the (i+1)'th character from the reference point of the i'th. The last increment may be needed to update current position.

These increments, if specified, set the widths of each character.

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCharStringPosAt - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_CHAR_POS_OPTIONS (0x204E)
An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_COORDINATE_OVERFLOW (0x2014)
An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCharStringPosAt - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear

GpiQueryCharStringPosAt - Example Code

This example uses the GpiQueryCharStringPosAt function to determine the location of each character in the string. Vector increments are not used.

```
#define INCL_GPIPRIMITIVES      /* GPI primitive functions      */
#include <os2.h>

HPS hps;                /* presentation space handle            */
POINTL ptlStart = { 100, 100 };
POINTL aptl[12];

GpiQueryCharStringPosAt(hps,  /* presentation-space handle      */
    &ptlStart,                /* starting point for string      */
    0L,                       /* do not use vector increments   */
    11L,                      /* 11 characters in string        */
    "This string",            /* character string               */
    NULL,                     /* no vector increments           */
    aptl);                    /* array of structures for points */
```

GpiQueryCharStringPosAt - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions

GpiQueryClipBox

GpiQueryClipBox - Syntax

This function returns the dimensions of the tightest rectangle which completely encloses the intersection of *all* the clipping definitions.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;          /*  Presentation-space handle. */
PRECTL    prclBound;    /*  Bounding rectangle. */
LONG      lComplexity;  /*  Complexity and error indicators. */

lComplexity = GpiQueryClipBox(hps, prclBound);
```

GpiQueryClipBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryClipBox Parameter - prclBound

prclBound (PRECTL) - output
Bounding rectangle.

The coordinates of the bounding rectangle, in world coordinates.

GpiQueryClipBox Return Value - lComplexity

lComplexity (LONG) - returns
Complexity and error indicators.

RGN_NULL
Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiQueryClipBox - Parameters

hps (HPS) - input
: Presentation-space handle.

prclBound (PRECTL) - output
: Bounding rectangle.

The coordinates of the bounding rectangle, in world coordinates.

lComplexity (LONG) - returns
: Complexity and error indicators.

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiQueryClipBox - Remarks

The clipping definitions include the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

Points on the borders of the rectangle returned are considered to be included within the rectangle. If the intersection is null, the rectangle returned has the right boundary less than the left, and the top boundary less than the bottom.

GpiQueryClipBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_COORDINATE_OVERFLOW (0x2014)
: An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or

too large.

GpiQueryClipBox - Example Code

This example uses GpiQueryClipBox to return the dimensions of the tightest rectangle which completely encloses the intersection of all the clipping definitions. The example queries the clip box and, if a rectangular region is returned, assigns the x-coordinate of the lower left hand corner of the clip box region to a variable.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>

LONG  lComplexity;      /* complexity/error indicator           */
HPS    hps;             /* Presentation-space handle            */
RECTL prclBound;        /* bounding rectangle                   */
LONG   lLwrLftxCoord;   /* lower left x coordinate of clip box  */

lComplexity = GpiQueryClipBox(hps, &prclBound);

/* if returned region is a rectangle, assign lower left x coordinate */
if (lComplexity == RGN_RECT)
   lLwrLftxCoord = prclBound.xLeft;
```

GpiQueryClipBox - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryClipRegion

GpiQueryClipRegion - Syntax

This function returns the handle of the currently selected clip region.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
HRGN    hrgn;  /*  Clip-region handle (if any). */

hrgn = GpiQueryClipRegion(hps);
```

GpiQueryClipRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryClipRegion Return Value - hrgn

hrgn (HRGN) - returns
Clip-region handle (if any).

NULLHANDLE
Null handle (no region is selected)
HRGN_ERROR
Error
Otherwise
Clip region handle.

GpiQueryClipRegion - Parameters

hps (HPS) - input
Presentation-space handle.

hrgn (HRGN) - returns
Clip-region handle (if any).

NULLHANDLE
Null handle (no region is selected)
HRGN_ERROR
Error
Otherwise
Clip region handle.

GpiQueryClipRegion - Remarks

If there is no currently selected clip region, a null handle is returned.

GpiQueryClipRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryClipRegion - Example Code

This example uses GpiQueryClipRegion to return the handle of the currently selected clip region.

```
#define INCL_GPIREGIONS         /* Region functions             */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
HRGN   hrgn;            /* clip region handle                  */

hrgn = GpiQueryClipRegion(hps);
```

GpiQueryClipRegion - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryColor

GpiQueryColor - Syntax

This function returns the current value of the (character) color attribute, as set by the GpiSetColor call.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lColor;  /*  Color attribute. */

lColor = GpiQueryColor(hps);
```

GpiQueryColor Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryColor Return Value - lColor

lColor (LONG) - returns
Color attribute.

If the presentation space is in RGB mode, *lColor* is an RGB value for the background. If the presentation space is in index mode (as set through GpiCreateLogColorTable), *lColor* is a CLR_* index value.

CLR_ERROR
Error
CLR_DEFAULT
Default
Otherwise
Color index.

GpiQueryColor - Parameters

hps (HPS) - input
Presentation-space handle.

lColor (LONG) - returns
Color attribute.

If the presentation space is in RGB mode, *lColor* is an RGB value for the background. If the presentation space is in index mode (as set through GpiCreateLogColorTable), *lColor* is a CLR_* index value.

CLR_ERROR
Error
CLR_DEFAULT
Default
Otherwise
Color index.

GpiQueryColor - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Valid CLR_* values are listed in the following list:

CLR_BACKGROUND (0L)
CLR_BLUE (1L)
CLR_RED (2L)
CLR_PINK (3L)
CLR_GREEN (4L)
CLR_CYAN (5L)
CLR_YELLOW (6L)
CLR_NEUTRAL (7L)
CLR_DARKGRAY (8L)
CLR_DARKBLUE (9L)
CLR_DARKRED (10L)
CLR_DARKPINK (11L)
CLR_DARKGREEN (12L)
CLR_DARKCYAN (13L)
CLR_BROWN (14L)
CLR_PALEGRAY (15L)

GpiQueryColor - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryColor - Example Code

This example uses GpiQueryColor to return the current value of the (character) color attribute, then sets the color to red by calling GpiSetColor. When finished with red, the color is set back to its original value.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

LONG   lColor;          /* current character color (or error)   */
HPS    hps;             /* Presentation-space handle            */

/* query current color */
lColor = GpiQueryColor(hps);

/* set color to red */
GpiSetColor(hps, CLR_RED);
 .
 .
/* restore to original color */
GpiSetColor(hps, lColor);
```

GpiQueryColor - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryColorData

GpiQueryColorData - Syntax

Information about the current logical color table or the selected palette is returned by this function.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;      /*  Presentation-space handle. */
LONG     lCount;   /*  Number of elements supplied in alArray. */
PLONG    alArray;  /*  Array. */
BOOL     rc;       /*  Success indicator. */

rc = GpiQueryColorData(hps, lCount, alArray);
```

GpiQueryColorData Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryColorData Parameter - lCount

lCount (LONG) - input
Number of elements supplied in *alArray*.

It must be between 0 and 4.

GpiQueryColorData Parameter - alArray

alArray (PLONG) - output
Array.

On return this array contains:

alArray[QCD_LCT_FORMAT]
Format of loaded color table if any. One of the following values is returned:

LCOLF_DEFAULT	Default color table is in force.
LCOLF_INDRGB	Color table loaded which provides translation from index to RGB.
LCOLF_RGB	Color index = RGB.
LCOLF_PALETTE	Palette is selected.

alArray[QCD_LCT_LOINDEX]
Smallest color index in the color table or palette; always zero for color tables.

alArray[QCD_LCT_HIINDEX]
Largest color index in the color table or palette; never less than 15 for color tables.

alArray[QCD_LCT_OPTIONS]
Color table or palette option. Zero or more of the following are returned:

LCOL_PURECOLOR	No color dithering (color table or selected palette).
LCOL_OVERRIDE_DEFAULT_COLORS	Override for applications that need the full hardware palette (selected palette only).

The array elements are numbered consecutively, starting with alArray. The element number constants start with 0.

Information is returned only for the number of elements supplied. Any extra elements supplied, beyond those described above, are set to zero by the system.

GpiQueryColorData Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryColorData - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of elements supplied in *alArray* .

It must be between 0 and 4.

alArray (PLONG) - output
Array.

On return this array contains:

alArray[QCD_LCT_FORMAT]
Format of loaded color table if any. One of the following values is returned:

LCOLF_DEFAULT	Default color table is in force.
LCOLF_INDRGB	Color table loaded which provides translation from index to RGB.
LCOLF_RGB	Color index = RGB.
LCOLF_PALETTE	Palette is selected.

alArray[QCD_LCT_LOINDEX]
Smallest color index in the color table or palette; always zero for color tables.

alArray[QCD_LCT_HIINDEX]
Largest color index in the color table or palette; never less than 15 for color tables.

alArray[QCD_LCT_OPTIONS]
Color table or palette option. Zero or more of the following are returned:

LCOL_PURECOLOR	No color dithering (color table or selected palette).
LCOL_OVERRIDE_DEFAULT_COLORS	Override for applications that need the full hardware palette (selected palette only).

The array elements are numbered consecutively, starting with alArray. The element number constants start with 0.

Information is returned only for the number of elements supplied. Any extra elements supplied, beyond those described above, are set to zero by the system.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryColorData - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

GpiQueryColorData - Example Code

This example uses the GpiQueryColorData function to retrieve the smallest color-table index. The GpiQueryLogColorTable function is then used to retrieve the RGB color value for this index.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

HPS hps;                /* presentation space handle            */
LONG alData[3];         /* information array                    */
LONG alColor[1];        /* information array                    */

GpiQueryColorData(hps, 3L, alData);
GpiQueryLogColorTable(hps, 0L, alData[QCD_LCT_LOINDEX],
                                        1L, alColor);
```

GpiQueryColorData - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryColorIndex

GpiQueryColorIndex - Syntax

This function returns the color index of the device color that is closest to the specified RGB color representation for the device connected to the specified presentation space.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
ULONG    flOptions;  /*  Options. */
LONG     lRgbColor;  /*  Specifies a color in RGB terms. */
LONG     lIndex;     /*  Color index providing closest match to the specified color. */

lIndex = GpiQueryColorIndex(hps, flOptions,
           lRgbColor);
```

GpiQueryColorIndex Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryColorIndex Parameter - flOptions

flOptions (ULONG) - input
Options.

Reserved, and must be zero.

GpiQueryColorIndex Parameter - lRgbColor

lRgbColor (LONG) - input
Specifies a color in RGB terms.

GpiQueryColorIndex Return Value - lIndex

lIndex (LONG) - returns
Color index providing closest match to the specified color.

>=0
Color index
- CLR_BACKGROUND (0L)
- CLR_BLUE (1L)
- CLR_RED (2L)
- CLR_PINK (3L)
- CLR_GREEN (4L)
- CLR_CYAN (5L)
- CLR_YELLOW (6L)
- CLR_NEUTRAL (7L)
- CLR_DARKGRAY (8L)
- CLR_DARKBLUE (9L)
- CLR_DARKRED (10L)
- CLR_DARKPINK (11L)
- CLR_DARKGREEN (12L)
- CLR_DARKCYAN (13L)
- CLR_BROWN (14L)
- CLR_PALEGRAY (15L)

GPI_ALTERROR
Error.

GpiQueryColorIndex - Parameters

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Options.

Reserved, and must be zero.

lRgbColor (LONG) - input
Specifies a color in RGB terms.

lIndex (LONG) - returns
Color index providing closest match to the specified color.

>=0
Color index.
- CLR_BACKGROUND (0L)
- CLR_BLUE (1L)
- CLR_RED (2L)
- CLR_PINK (3L)
- CLR_GREEN (4L)
- CLR_CYAN (5L)
- CLR_YELLOW (6L)
- CLR_NEUTRAL (7L)
- CLR_DARKGRAY (8L)
- CLR_DARKBLUE (9L)
- CLR_DARKRED (10L)
- CLR_DARKPINK (11L)
- CLR_DARKGREEN (12L)
- CLR_DARKCYAN (13L)
- CLR_BROWN (14L)
- CLR_PALEGRAY (15L)

GPI_ALTERROR
Error.

GpiQueryColorIndex - Remarks

If an RGB logical color table has been loaded, this call returns the same RGB color that is passed to it.

GpiQueryColorIndex - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COLOR_OPTIONS (0x2057)
An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_RGBCOLOR (0x20C3)
An invalid rgb color parameter was specified with GpiQueryNearestColor or GpiQueryColor.

GpiQueryColorIndex - Example Code

This example uses GpiQueryColorIndex to return the color index of the device color that is closest to the specified RGB color representation for the device connected to the specified presentation space.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

LONG   lIndex;          /* closest match color index            */
HPS    hps;             /* Presentation-space handle            */
ULONG  ulOptions;       /* options                              */
LONG   lRgbColor;       /* color in RGB terms                   */

/* reserved; set to 0 */
ulOptions = 0L;

/* color to find index for (r=137, g=136, b= 135 */
lRgbColor = (137*65536) + (136*256) + 135;

lIndex = GpiQueryColorIndex(hps, ulOptions, lRgbColor);
```

GpiQueryColorIndex - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryCp

GpiQueryCp - Syntax

This function returns the currently selected graphics code page.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;         /*  Presentation-space handle. */
ULONG    ulCodePage;  /*  Code page. */

ulCodePage = GpiQueryCp(hps);
```

GpiQueryCp Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCp Return Value - ulCodePage

ulCodePage (ULONG) - returns
Code page.

GPI_ERROR
Error
Otherwise
Code page.

GpiQueryCp - Parameters

hps (HPS) - input
Presentation-space handle.

ulCodePage (ULONG) - returns
Code page.

GPI_ERROR
Error
Otherwise
Code page.

GpiQueryCp - Remarks

The code page identity returned is the one that is set by GpiSetCp (or defaulted when the presentation space is first created). This is the code page of the default font, not the currently-selected font, found from GpiQueryFontMetrics.

GpiQueryCp - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryCp - Example Code

This example uses GpiQueryCp to return the currently selected graphics code page.

```
#define INCL_GPILCIDS           /* Font functions               */
#include <os2.h>

ULONG  ulCodePage;      /* code page (or error)                 */
HPS    hps;             /* Presentation-space handle            */

ulCodePage = GpiQueryCp(hps);
```

GpiQueryCp - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryCurrentPosition

GpiQueryCurrentPosition - Syntax

This function returns the value of current position.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlPoint;  /*  Current position. */
BOOL       rc;         /*  Success indicator. */

rc = GpiQueryCurrentPosition(hps, pptlPoint);
```

GpiQueryCurrentPosition Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryCurrentPosition Parameter - pptlPoint

pptlPoint (PPOINTL) - output
Current position.

GpiQueryCurrentPosition Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCurrentPosition - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - output
Current position.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryCurrentPosition - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryCurrentPosition - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryCurrentPosition - Example Code

This example uses GpiQueryCurrentPosition to return the value of the current position and assigns the x coordinate to a variable.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
POINTL pptlPoint;       /* current position                     */
LONG   lxCoord;         /* current position x coordinate        */

fSuccess = GpiQueryCurrentPosition(hps, &pptlPoint);

if (fSuccess == TRUE)
   lxCoord = pptlPoint.x;
```

GpiQueryCurrentPosition - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryDefArcParams

GpiQueryDefArcParams - Syntax

This function returns the default values of the arc parameters, as set by the GpiSetDefArcParams function.

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS           hps;             /*  Presentation-space handle. */
PARCPARAMS    parcpArcParams;  /*  Default arc parameters. */
BOOL          rc;              /*  Success indicator. */

rc = GpiQueryDefArcParams(hps, parcpArcParams);
```

GpiQueryDefArcParams Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDefArcParams Parameter - parcpArcParams

parcpArcParams (PARCPARAMS) - output
Default arc parameters.

GpiQueryDefArcParams Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryDefArcParams - Parameters

hps (HPS) - input
Presentation-space handle.

parcpArcParams (PARCPARAMS) - output
Default arc parameters.

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryDefArcParams - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryDefArcParams - Example Code

This example uses GpiQueryDefArcParams to return the default values of the arc parameters, as set by the GpiSetDefArcParams call, and assign a variable to the P coefficient if the query succeeds.

```
#define INCL_GPIDEFAULTS        /* Default functions            */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
ARCPARAMS  parcpArcParams; /* Arc parameters                    */
LONG  lPcoefficient;    /* p coefficient of arc definition      */

fSuccess = GpiQueryDefArcParams(hps, &parcpArcParams);

/* if successful, assign value of P coefficient */
if (fSuccess == TRUE)
   lPcoefficient = parcpArcParams.lP;
```

GpiQueryDefArcParams - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryDefAttrs

GpiQueryDefAttrs - Syntax

This function returns default attribute values for the specified primitive type.

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lPrimType;   /*  Primitive type. */
ULONG      flAttrMask;  /*  Attributes mask. */
PBUNDLE    ppbunAttrs;  /*  Attributes. */
BOOL       rc;          /*  Success indicator. */

rc = GpiQueryDefAttrs(hps, lPrimType, flAttrMask,
       ppbunAttrs);
```

GpiQueryDefAttrs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDefAttrs Parameter - lPrimType

lPrimType (LONG) - input
Primitive type.

This is the type of primitive for which default attribute values are to be queried, as follows:

PRIM_LINE
: Line and arc primitives

PRIM_CHAR
: Character primitives

PRIM_MARKER
: Marker primitives

PRIM_AREA
: Area primitives

PRIM_IMAGE
: Image primitives.

GpiQueryDefAttrs Parameter - flAttrMask

flAttrMask (ULONG) - input
Attributes mask.

Each flag that is set indicates that the default value of the corresponding attribute is to be returned in the *ppbunAttrs* buffer. If all flags

in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

Line attributes:

LBB_COLOR	Line color
LBB_MIX_MODE	Line mix
LBB_WIDTH	Line width
LBB_GEOM_WIDTH	Geometric line width
LBB_TYPE	Line type
LBB_END	Line end
LBB_JOIN	Line join.

Character attributes:

CBB_COLOR	Character color
CBB_BACK_COLOR	Character background color
CBB_MIX_MODE	Character mix
CBB_BACK_MIX_MODE	Character background mix
CBB_SET	Character set
CBB_MODE	Character mode
CBB_BOX	Character box
CBB_ANGLE	Character angle
CBB_SHEAR.	Character shear
CBB_DIRECTION	Character direction
CBB_EXTRA	Character extra
CBB_BREAK_EXTRA	Character break extra

Marker attributes:

MBB_COLOR	Marker color
MBB_BACK_COLOR	Marker background color
MBB_MIX_MODE	Marker mix
MBB_BACK_MIX_MODE	Marker background mix
MBB_SET	Marker set
MBB_SYMBOL	Marker symbol
MBB_BOX	Marker box.

Pattern attributes (areas):

ABB_COLOR	Area color
ABB_BACK_COLOR	Area background color
ABB_MIX_MODE	Area mix
ABB_BACK_MIX_MODE	Area background mix
ABB_SET	Pattern set
ABB_SYMBOL	Pattern symbol
ABB_REF_POINT	Pattern reference point.

Image attributes:

IBB_COLOR	Image color
IBB_BACK_COLOR	Image background color
IBB_MIX_MODE	Image mix
IBB_BACK_MIX_MODE	Image background mix.

GpiQueryDefAttrs Parameter - ppbunAttrs

ppbunAttrs (PBUNDLE) - output
Attributes.

ppbunAttrs is a buffer in which is returned the default value of each attribute for which the *flAttrMask* flag is set, in the order specified in GpiSetAttrs for the particular primitive type.

Only data for attributes for which the appropriate flag in *flAttrMask* is set is updated, so *ppbunAttrs* need only be large enough for the highest offset attribute to be returned (see GpiSetAttrs).

GpiQueryDefAttrs Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryDefAttrs - Parameters

hps (HPS) - input
Presentation-space handle.

lPrimType (LONG) - input
Primitive type.

This is the type of primitive for which default attribute values are to be queried, as follows:

PRIM_LINE
: Line and arc primitives

PRIM_CHAR
: Character primitives

PRIM_MARKER
: Marker primitives

PRIM_AREA
: Area primitives

PRIM_IMAGE
: Image primitives.

flAttrMask (ULONG) - input
Attributes mask.

Each flag that is set indicates that the default value of the corresponding attribute is to be returned in the *ppbunAttrs* buffer. If all flags in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

Line attributes:

LBB_COLOR	Line color
LBB_MIX_MODE	Line mix
LBB_WIDTH	Line width
LBB_GEOM_WIDTH	Geometric line width
LBB_TYPE	Line type
LBB_END	Line end
LBB_JOIN	Line join.

Character attributes:

CBB_COLOR	Character color
CBB_BACK_COLOR	Character background color
CBB_MIX_MODE	Character mix
CBB_BACK_MIX_MODE	Character background mix
CBB_SET	Character set
CBB_MODE	Character mode
CBB_BOX	Character box
CBB_ANGLE	Character angle
CBB_SHEAR.	Character shear
CBB_DIRECTION	Character direction

CBB_EXTRA	Character extra
CBB_BREAK_EXTRA	Character break extra

Marker attributes:

MBB_COLOR	Marker color
MBB_BACK_COLOR	Marker background color
MBB_MIX_MODE	Marker mix
MBB_BACK_MIX_MODE	Marker background mix
MBB_SET	Marker set
MBB_SYMBOL	Marker symbol
MBB_BOX	Marker box.

Pattern attributes (areas):

ABB_COLOR	Area color
ABB_BACK_COLOR	Area background color
ABB_MIX_MODE	Area mix
ABB_BACK_MIX_MODE	Area background mix
ABB_SET	Pattern set
ABB_SYMBOL	Pattern symbol
ABB_REF_POINT	Pattern reference point.

Image attributes:

IBB_COLOR	Image color
IBB_BACK_COLOR	Image background color
IBB_MIX_MODE	Image mix
IBB_BACK_MIX_MODE	Image background mix.

ppbunAttrs (PBUNDLE) - output
Attributes.

ppbunAttrs is a buffer in which is returned the default value of each attribute for which the *flAttrMask* flag is set, in the order specified in GpiSetAttrs for the particular primitive type.

Only data for attributes for which the appropriate flag in *flAttrMask* is set is updated, so *ppbunAttrs* need only be large enough for the highest offset attribute to be returned (see GpiSetAttrs).

rc (BOOL) - returns
Success indicator.

TRUE	Successful completion
FALSE	Error occurred.

GpiQueryDefAttrs - Remarks

The parameters returned by this function may be used to reinstate exactly the same default attribute values as are queried, using GpiSetDefAttrs.

GpiQueryDefAttrs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PRIMITIVE_TYPE (0x20B9)
An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.

PMERR_UNSUPPORTED_ATTR (0x2109)
An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.

GpiQueryDefAttrs - Example Code

This example uses GpiQueryDefAttrs to return the default color and mix attribute values for the primitive line and arc types and, if successful, uses the values to reinstate the default attributes via the DosSetDefAttrs API.

```
#define INCL_GPIDEFAULTS        /* Default functions           */
#define INCL_GPIPRIMITIVES
#include <os2.h>

BOOL       fSuccess;   /* success indicator                   */
HPS        hps;        /* Presentation-space handle           */
LONG       lPrimType;  /* primitive type                      */
ULONG      flAttrMask; /* attributes mask                     */
LINEBUNDLE ppbunAttrs; /* Attributes                          */

/* request line/arc primitive values */
lPrimType = PRIM_LINE;

/* request values for color, mix attributes */
flAttrMask = LBB_COLOR | LBB_MIX_MODE;

fSuccess = GpiQueryDefAttrs(hps, lPrimType, flAttrMask,
                            &ppbunAttrs);

/* if successful, reinstate default color and mix attributes  */
if (fSuccess == TRUE)
   fSuccess = GpiSetDefAttrs(hps, lPrimType, flAttrMask,
                             &ppbunAttrs);
```

GpiQueryDefAttrs - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryDefaultViewMatrix

GpiQueryDefaultViewMatrix - Syntax

This function returns the current default viewing transform; see GpiSetDefaultViewMatrix.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
LONG         lCount;       /*  Number of elements. */
PMATRIXLF    pmatlfArray;  /*  Transform matrix. */
BOOL         rc;           /*  Success indicator. */

rc = GpiQueryDefaultViewMatrix(hps, lCount,
       pmatlfArray);
```

GpiQueryDefaultViewMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDefaultViewMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements to be returned in *pmatlfArray*; must be in the range 0 through 9. If 0 is specified, no matrix elements are returned.

GpiQueryDefaultViewMatrix Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - output
Transform matrix.

An array into which the elements of the default viewing transform matrix are returned.

GpiQueryDefaultViewMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiQueryDefaultViewMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of elements.

The number of elements to be returned in *pmatlfArray*; must be in the range 0 through 9. If 0 is specified, no matrix elements are returned.

pmatlfArray (PMATRIXLF) - output
Transform matrix.

An array into which the elements of the default viewing transform matrix are returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryDefaultViewMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

GpiQueryDefaultViewMatrix - Example Code

This example uses GpiQueryDefaultViewMatrix to return the default viewing transform and, if successful, defines - via GpiSetDefaultViewMatrix - the returned value as the new default transform.

```
#define INCL_GPITRANSFORMS      /* Transform functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
LONG  lCount;           /* number of elements                   */
MATRIXLF  pmatlfArray;  /* transform matrix                     */
LONG     lOptions;      /* set default options                  */
```

```
lCount = 1; /* examine only first element of transform matrix */

fSuccess = GpiQueryDefaultViewMatrix(hps, lCount, &pmatlfArray);

/* set default to returned transform */
if (fSuccess == TRUE)
   {
   lOptions = TRANSFORM_REPLACE;
   fSuccess = GpiSetDefaultViewMatrix(hps, lCount, &pmatlfArray,
                                      lOptions);
   }
```

GpiQueryDefaultViewMatrix - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryDefCharBox

GpiQueryDefCharBox - Syntax

This function returns the size of the default graphics character box in world coordinates.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
PSIZEL    psizlSize;  /*  Default character-box size. */
BOOL      rc;         /*  Success indicator. */

rc = GpiQueryDefCharBox(hps, psizlSize);
```

GpiQueryDefCharBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDefCharBox Parameter - psizlSize

psizlSize (PSIZEL) - output
Default character-box size.

GpiQueryDefCharBox Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryDefCharBox - Parameters

hps (HPS) - input
Presentation-space handle.

psizlSize (PSIZEL) - output
Default character-box size.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryDefCharBox - Remarks

The values returned are the same as the initial default value of the character-box attribute. See GpiSetCharBox for further information.

GpiQueryDefCharBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

GpiQueryDefCharBox - Example Code

This example uses GpiQueryDefCharBox to query the initial size of the default graphics character box in world coordinates and, if the query succeeds, resets the current size back to this initial default value via GpiSetCharBox (note the required transformation from LONG to FIXED using the MAKEFIXED macro).

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
SIZEL  psizlSize;       /* default character-box size           */
SIZEF  psizfxSize;      /* new character-box size               */

fSuccess = GpiQueryDefCharBox(hps, &psizlSize);

/* if successful, set current box size to initial default value */
if (fSuccess == TRUE)
   {
   psizfxSize.cx = MAKEFIXED(psizlSize.cx,0x0000);
   psizfxSize.cy = MAKEFIXED(psizlSize.cy,0x0000);
   GpiSetCharBox(hps, &psizfxSize);
   }
```

GpiQueryDefCharBox - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryDefTag

GpiQueryDefTag - Syntax

This function returns the default value of the tag identifier, as set by the GpiSetDefTag function.

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;    /*  Presentation-space handle. */
PLONG    plTag;  /*  Default tag identifier. */
BOOL     rc;     /*  Success indicator. */

rc = GpiQueryDefTag(hps, plTag);
```

GpiQueryDefTag Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDefTag Parameter - plTag

plTag (PLONG) - output
Default tag identifier.

GpiQueryDefTag Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryDefTag - Parameters

hps (HPS) - input
Presentation-space handle.

plTag (PLONG) - output
Default tag identifier.

rc (BOOL) - returns
Success indicator.

TRUE

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryDefTag - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQueryDefTag - Example Code

This example uses GpiQueryDefTag to return the default value of the tag identifier, as set by the GpiSetDefTag call.

```
#define INCL_GPIDEFAULTS        /* Default functions            */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
LONG  plTag;            /* default tag identifier               */

fSuccess = GpiQueryDefTag(hps, &plTag);
```

GpiQueryDefTag - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryDefViewingLimits

GpiQueryDefViewingLimits - Syntax

This function returns the default value of the viewing limits, as set by the GpiSetDefViewingLimits function.

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;         /*  Presentation-space handle. */
PRECTL    prclLimits;  /*  Default viewing limits. */
BOOL      rc;          /*  Success indicator. */

rc = GpiQueryDefViewingLimits(hps, prclLimits);
```

GpiQueryDefViewingLimits Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDefViewingLimits Parameter - prclLimits

prclLimits (PRECTL) - output
Default viewing limits.

GpiQueryDefViewingLimits Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryDefViewingLimits - Parameters

hps (HPS) - input
Presentation-space handle.

prclLimits (PRECTL) - output
Default viewing limits.

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryDefViewingLimits - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryDefViewingLimits - Example Code

This example uses GpiQueryDefViewingLimits to return the default value of the viewing limits, as set by the GpiSetDefViewingLimits and, if the query succeeds, assigns a variable to the x coordinate of the lower left hand corner of the viewing limits rectangle.

```
#define INCL_GPIDEFAULTS        /* Default functions            */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
RECTL prclLimits;       /* default viewing limits               */
LONG   lLwrLftxCoord;   /* lower left x coordinate of limit     */

fSuccess = GpiQueryDefViewingLimits(hps, &prclLimits);

/* if successful, assign lower left x coordinate of viewing limit */
if (fSuccess == TRUE)
   lLwrLftxCoord = prclLimits.xLeft;
```

GpiQueryDefViewingLimits - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryDevice

GpiQueryDevice - Syntax

This function returns the handle of the currently associated device context.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS    hps;  /*  Presentation-space handle. */
HDC    hdc;  /*  Device-context handle. */

hdc = GpiQueryDevice(hps);
```

GpiQueryDevice Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDevice Return Value - hdc

hdc (HDC) - returns
Device-context handle.

HDC_ERROR
Error
NULLHANDLE
No device context is currently associated
Otherwise
Device context handle.

GpiQueryDevice - Parameters

hps (HPS) - input
Presentation-space handle.

hdc (HDC) - returns
Device-context handle.

HDC_ERROR
Error
NULLHANDLE
No device context is currently associated
Otherwise
Device context handle.

GpiQueryDevice - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_INV_HDC (0x207C)
An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

GpiQueryDevice - Example Code

This example uses the GpiQueryDevice function to retrieve a device-context handle for the presentation space of the desktop window. The handle is used in the DevQueryCaps function to determine the width and height of the Presentation Manager screen.

```
#define INCL_GPICONTROL         /* GPI control Functions        */
#define INCL_WINWINDOWMGR       /* Window Manager Functions     */
#define INCL_DEV                /* Device Function definitions  */
#include <os2.h>

HPS hps;                /* presentation space handle            */
HDC hdc;                /* device context handle                */
LONG lWidth, lHeight;

hps = WinGetScreenPS(HWND_DESKTOP);
hdc = GpiQueryDevice(hps);
DevQueryCaps(hdc, CAPS_WIDTH, 1L, &lWidth);
DevQueryCaps(hdc, CAPS_HEIGHT, 1L, &lHeight);
```

GpiQueryDevice - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryDeviceBitmapFormats

GpiQueryDeviceBitmapFormats - Syntax

This function returns the formats of bit maps supported internally by the device driver.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;      /*  Presentation-space handle. */
LONG     lCount;   /*  Number of elements. */
PLONG    alArray;  /*  Data array. */
BOOL     rc;       /*  Success indicator. */

rc = GpiQueryDeviceBitmapFormats(hps, lCount,
       alArray);
```

GpiQueryDeviceBitmapFormats Parameter - hps

hps (HPS) - input
Presentation-space handle.

The associated device context defines the class of device for which formats are required. This must be either a memory device context or a device context for a device that supports raster operations.

GpiQueryDeviceBitmapFormats Parameter - lCount

lCount (LONG) - input
Number of elements.

Number of elements in *alArray* (must be a positive even number). For the complete set of formats returned, the value of this parameter must be at least double the number of device formats returned by DevQueryCaps

GpiQueryDeviceBitmapFormats Parameter - alArray

alArray (PLONG) - output
Data array.

Array of elements that, on return, is set to pairs of (*cPlanes* , *cBitCount*)elements (see BITMAPINFOHEADER) for each supported format in turn. Any unused elements are set to 0.

GpiQueryDeviceBitmapFormats Return Value - rc

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryDeviceBitmapFormats - Parameters

hps (HPS) - input
: Presentation-space handle.

 The associated device context defines the class of device for which formats are required. This must be either a memory device context or a device context for a device that supports raster operations.

lCount (LONG) - input
: Number of elements.

 Number of elements in *alArray* (must be a positive even number). For the complete set of formats returned, the value of this parameter must be at least double the number of device formats returned by DevQueryCaps

alArray (PLONG) - output
: Data array.

 Array of elements that, on return, is set to pairs of (*cPlanes* , *cBitCount*)elements (see BITMAPINFOHEADER) for each supported format in turn. Any unused elements are set to 0.

rc (BOOL) - returns
: Success indicator.

 TRUE
 : Successful completion

 FALSE
 : Error occurred.

GpiQueryDeviceBitmapFormats - Remarks

An application can create, set, and query bit maps using any of the standard formats. Internally, however, these are converted by the device driver into one of the device internal formats if necessary. This is normally a smaller set than the standard set of bit-map formats.

The number of device bit-map formats can be found with DevQueryCaps (CAPS_BITMAP_FORMATS).

The first pair of (*cPlanes* , *cBitCount*) elements returned most closely matches the device.

This function must not be issued when there is no device context associated with the presentation space.

GpiQueryDeviceBitmapFormats - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

GpiQueryDeviceBitmapFormats - Example Code

This example uses the GpiQueryDeviceBitmapFormats function to retrieve bit-map formats for the screen and creates a screen-compatible bit map with GpiCreateBitmap.

```
#define INCL_GPIBITMAPS         /* GPI Bit-map functions         */
#include <os2.h>

HPS          hps;       /* Target presentation-space handle     */
LONG         lFormats[24];/* Formats supported by the device    */
HBITMAP      hbm;       /* Bit-map handle                       */
PBYTE        pb;        /* Bit-map image data                   */
BITMAPINFO2  pbmInfo;   /* Bit-map information table            */

/* Get screen supportable formats */
GpiQueryDeviceBitmapFormats(hps, 24L, lFormats);


/***************************
 * set bitmapinfo structure *
 ***************************/
pbmInfo.cbFix = 16L;
pbmInfo.cx = 100L;
pbmInfo.cy = 100L;
pbmInfo.cPlanes = (USHORT) lFormats[0] ;
pbmInfo.cBitCount = (USHORT) lFormats[1];

/* create bit map and return handle */
hbm = GpiCreateBitmap(hps,       /* presentation space         */
                  (PBITMAPINFOHEADER2)&pbmInfo,
                                 /* bit-map information header */
                  CBM_INIT,      /* initialize the bit map     */
                  pb,            /* bit-map data               */
                  &pbmInfo);     /* bit-map information table  */
```

GpiQueryDeviceBitmapFormats - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryDrawControl

GpiQueryDrawControl - Syntax

This function returns a drawing control as set by GpiSetDrawControl.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lControl;  /*  Control whose value is to be returned. */
LONG    lValue;    /*  Value of the control. */

lValue = GpiQueryDrawControl(hps, lControl);
```

GpiQueryDrawControl Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDrawControl Parameter - lControl

lControl (LONG) - input
Control whose value is to be returned.

DCTL_ERASE
Erase before draw
DCTL_DISPLAY
Display
DCTL_BOUNDARY
Accumulate boundary data
DCTL_DYNAMIC
Draw dynamic segments
DCTL_CORRELATE
Correlate.

GpiQueryDrawControl Return Value - lValue

lValue (LONG) - returns
Value of the control.

(See GpiSetDrawControl for details):

DCTL_OFF
Off
DCTL_ON

On
DCTL_ERROR
Error.

GpiQueryDrawControl - Parameters

hps (HPS) - input
Presentation-space handle.

lControl (LONG) - input
Control whose value is to be returned.

DCTL_ERASE
Erase before draw
DCTL_DISPLAY
Display
DCTL_BOUNDARY
Accumulate boundary data
DCTL_DYNAMIC
Draw dynamic segments
DCTL_CORRELATE
Correlate.

lValue (LONG) - returns
Value of the control.

(See GpiSetDrawControl for details):

DCTL_OFF
Off
DCTL_ON
On
DCTL_ERROR
Error.

GpiQueryDrawControl - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_DRAW_CONTROL (0x2063)
An invalid control parameter was specified with GpiSetDrawControl or GpiQueryDrawControl.

PMERR_INV_MICROPS_DRAW_CONTROL (0x20A0)
A draw control parameter was specified with GpiSetDrawControl that is invalid in a micro presentation space.

GpiQueryDrawControl - Example Code

This example uses GpiQueryDrawControl to return the value for the Display drawing control as set by GpiSetDrawControl.

```
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

LONG     lValue;        /* value of the control                 */
HPS    hps;             /* Presentation-space handle            */
LONG   lControl;        /* control value to be queried          */

/* ask for Display control value */
lControl = DCTL_DISPLAY;

lValue = GpiQueryDrawControl(hps, lControl);
```

GpiQueryDrawControl - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryDrawingMode

GpiQueryDrawingMode - Syntax

This function returns the current drawing mode, as set by GpiSetDrawingMode.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lMode;  /*  Drawing mode. */

lMode = GpiQueryDrawingMode(hps);
```

GpiQueryDrawingMode Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryDrawingMode Return Value - lMode

lMode (LONG) - returns
Drawing mode.

(See GpiSetDrawingMode for details):

>0
Drawing mode
DM_ERROR
Error.

GpiQueryDrawingMode - Parameters

hps (HPS) - input
Presentation-space handle.

lMode (LONG) - returns
Drawing mode.

(See GpiSetDrawingMode for details):

>0
Drawing mode
DM_ERROR
Error.

GpiQueryDrawingMode - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQueryDrawingMode - Example Code

This example uses GpiQueryDrawingMode to return the current drawing mode, as set by GpiSetDrawingMode.

```
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

LONG     lMode;         /* drawing mode                         */
```

```
HPS    hps;             /* Presentation-space handle            */

lMode = GpiQueryDrawingMode(hps);
```

GpiQueryDrawingMode - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryEditMode

GpiQueryEditMode - Syntax

This function returns the current editing mode, as set by GpiSetEditMode.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lMode;  /*  Current editing mode. */

lMode = GpiQueryEditMode(hps);
```

GpiQueryEditMode Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryEditMode Return Value - lMode

lMode (LONG) - returns

Current editing mode.

SEGEM_INSERT
: Insert mode

SEGEM_REPLACE
: Replace mode

SEGEM_ERROR
: Error.

GpiQueryEditMode - Parameters

hps (HPS) - input
: Presentation-space handle.

lMode (LONG) - returns
: Current editing mode.

SEGEM_INSERT
: Insert mode

SEGEM_REPLACE
: Replace mode

SEGEM_ERROR
: Error.

GpiQueryEditMode - Remarks

This function can be issued in any drawing mode.

GpiQueryEditMode - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQueryEditMode - Example Code

This example uses GpiQueryEditMode to return the current editing mode, as set by GpiSetEditMode.

```
#define INCL_GPISEGEDITING      /* Segment Editing functions    */
#include <os2.h>
```

```
LONG     lMode;         /* editing mode                        */
HPS    hps;             /* Presentation-space handle           */

lMode = GpiQueryEditMode(hps);
```

GpiQueryEditMode - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryElement

GpiQueryElement - Syntax

This function returns element content data.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;         /*  Presentation-space handle. */
LONG     lOff;        /*  Starting byte offset within the element. */
LONG     lMaxLength;  /*  Maximum length of data that can be returned. */
PBYTE    pbData;      /*  Element content data. */
LONG     lRetLength;  /*  Number of bytes returned. */

lRetLength = GpiQueryElement(hps, lOff, lMaxLength,
               pbData);
```

GpiQueryElement Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryElement Parameter - lOff

lOff (LONG) - input
Starting byte offset within the element.

GpiQueryElement Parameter - lMaxLength

lMaxLength (LONG) - input
Maximum length of data that can be returned.

GpiQueryElement Parameter - pbData

pbData (PBYTE) - output
Element content data.

An area of *lMaxLength* bytes in which the element content data is to be returned.

GpiQueryElement Return Value - lRetLength

lRetLength (LONG) - returns
Number of bytes returned.

>=0
Actual number of bytes returned
GPI_ALTERROR
Error.

GpiQueryElement - Parameters

hps (HPS) - input
Presentation-space handle.

lOff (LONG) - input
Starting byte offset within the element.

lMaxLength (LONG) - input
Maximum length of data that can be returned.

pbData (PBYTE) - output
Element content data.

An area of *lMaxLength* bytes in which the element content data is to be returned.

lRetLength (LONG) - returns
Number of bytes returned.

>=0
Actual number of bytes returned
GPI_ALTERROR
Error.

GpiQueryElement - Remarks

Returns the element content (or part of the element content) for the element to which the element pointer currently points.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress.

This function is not valid within an element bracket.

GpiQueryElement - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NO_CURRENT_ELEMENT (0x20E5)
An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open element.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

PMERR_INV_ELEMENT_OFFSET (0x206A)
An invalid off (offset) parameter was specified with GpiQueryElement.

PMERR_NO_CURRENT_SEG (0x20E6)
An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

GpiQueryElement - Example Code

This example uses the GpiQueryElement function to retrieve the graphics-order data for an element.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#include <os2.h>
```

```
HPS  hps;               /* presentation space handle            */
BYTE abElement[512];    /* element data buffer                  */

/* Move pointer to first element in segment. */

GpiSetElementPointer(hps, 1L);
GpiQueryElement(hps,    /* presentation space               */
    0L,                 /* start with first byte in element */
    512L,               /* copy no more than 512 bytes      */
    abElement);         /* buffer to receive data           */
```

GpiQueryElement - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryElementPointer

GpiQueryElementPointer - Syntax

This function returns the current element pointer.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lElement;  /*  Current element pointer. */

lElement = GpiQueryElementPointer(hps);
```

GpiQueryElementPointer Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryElementPointer Return Value - lElement

lElement (LONG) - returns
Current element pointer.

>=0
Current element pointer
GPI_ALTERROR
Error.

GpiQueryElementPointer - Parameters

hps (HPS) - input
Presentation-space handle.

lElement (LONG) - returns
Current element pointer.

>=0
Current element pointer
GPI_ALTERROR
Error.

GpiQueryElementPointer - Remarks

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress.

GpiQueryElementPointer - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_NO_CURRENT_SEG (0x20E6)
An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

GpiQueryElementPointer - Example Code

This example uses GpiQueryElementPointer to return the current element pointer after setting the Draw mode to retain and beginning a graphics segment named 1.

```
#define INCL_GPISEGEDITING      /* Segment Editing functions    */
#define INCL_GPICONTROL         /* Control functions            */
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

LONG     lElement;      /* current element pointer             */
HPS    hps;             /* Presentation-space handle           */

/* set the draw mode to retain and open the segment */
if (GpiSetDrawingMode(hps, DM_RETAIN) == TRUE &&
    GpiOpenSegment(hps, 1L) == TRUE)
   {
   lElement = GpiQueryElementPointer(hps);
   GpiCloseSegment(hps);  /* close the segment */
   }
```

GpiQueryElementPointer - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryElementType

GpiQueryElementType - Syntax

This function returns information about the element to which the element pointer currently points.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;         /*  Presentation-space handle. */
PLONG    plType;      /*  Element type. */
LONG     lLength;     /*  Data length. */
PSZ      pszData;     /*  Description of data buffer. */
LONG     lReqLength;  /*  Size of the data required to hold the element content. */

lReqLength = GpiQueryElementType(hps, plType,
               lLength, pszData);
```

GpiQueryElementType Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryElementType Parameter - plType

plType (PLONG) - output
Element type.

The element type can be system-defined or application-defined; see GpiElement and GpiBeginElement.

GpiQueryElementType Parameter - lLength

lLength (LONG) - input
Data length.

Length of the description data buffer.

GpiQueryElementType Parameter - pszData

pszData (PSZ) - output
Description of data buffer.

The description may be system-defined or application-defined; see GpiElement and GpiBeginElement. The string is null-terminated, even if it has to be truncated.

GpiQueryElementType Return Value - lReqLength

lReqLength (LONG) - returns
Size of the data required to hold the element content.

This can be used for a subsequent GpiQueryElement function.

>=0

Size of data

GPI_ALTERROR
Error.

GpiQueryElementType - Parameters

hps (HPS) - input
Presentation-space handle.

plType (PLONG) - output
Element type.

The element type can be system-defined or application-defined; see GpiElement and GpiBeginElement.

lLength (LONG) - input
Data length.

Length of the description data buffer.

pszData (PSZ) - output
Description of data buffer.

The description may be system-defined or application-defined; see GpiElement and GpiBeginElement. The string is null-terminated, even if it has to be truncated.

lReqLength (LONG) - returns
Size of the data required to hold the element content.

This can be used for a subsequent GpiQueryElement function.

>=0
Size of data

GPI_ALTERROR
Error.

GpiQueryElementType - Remarks

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is not valid in an element bracket.

GpiQueryElementType - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NO_CURRENT_ELEMENT (0x20E5)

An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open element.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

PMERR_NO_CURRENT_SEG (0x20E6)
An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

GpiQueryElementType - Example Code

This example uses the GpiQueryElementType function to retrieve the size of the current element. The size is used to retrieve the graphics-order data in the element.

```
#define INCL_GPISEGEDITING      /* GPI Segment Edit functions   */
#include <os2.h>

HPS  hps;               /* presentation space handle           */
BYTE abElement[512];
LONG cbElement;
LONG lType;

/* move pointer to first element in segment */

GpiSetElementPointer(hps, 1L);
cbElement = GpiQueryElementType(
    hps,      /* presentation space             */
    &lType,   /* variable to receive type       */
    0L,       /* copy zero bytes of description */
    NULL);    /* no buffer for description      */

GpiQueryElement(hps, 0L, cbElement, abElement);
```

GpiQueryElementType - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryFaceString

GpiQueryFaceString - Syntax

This function generates a compound face name for a font.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS             ps;               /*  Presentation-space handle. */
PSZ             FamilyName;       /*  Family name. */
PFACENAMEDESC   attrs;            /*  Face-name description. */
LONG            length;           /*  Length of CompoundFaceName buffer. */
PSZ             CompoundFaceName; /*  Compound face name. */
ULONG           cbRetLength;      /*  Length of the compound face name. */

cbRetLength = GpiQueryFaceString(ps, FamilyName,
               attrs, length, CompoundFaceName);
```

GpiQueryFaceString Parameter - ps

ps (HPS) - input
Presentation-space handle.

GpiQueryFaceString Parameter - FamilyName

FamilyName (PSZ) - input
Family name.

The family name of the font (for example, "Courier").

GpiQueryFaceString Parameter - attrs

attrs (PFACENAMEDESC) - input
Face-name description.

A structure that provides the characteristics of the required font. These characteristics are used to generate the compound face name.

GpiQueryFaceString Parameter - length

length (LONG) - input
Length of *CompoundFaceName* buffer.

The maximum length of the compound face name returned (including the trailing zero of the string).

Specify zero to find out how large the *CompoundFaceName* buffer needs to be.

GpiQueryFaceString Parameter - CompoundFaceName

CompoundFaceName (PSZ) - output
Compound face name.

The compound face name of the font.

GpiQueryFaceString Return Value - cbRetLength

cbRetLength (ULONG) - returns
Length of the compound face name.

GPI_ERROR
Error occurred
> 0
Length of the compound face-name string (including the trailing zero). This is the length of the complete string; if it is greater than *length*, the string returned in *CompoundFaceName* is truncated.

GpiQueryFaceString - Parameters

ps (HPS) - input
Presentation-space handle.

FamilyName (PSZ) - input
Family name.

The family name of the font (for example, "Courier").

attrs (PFACENAMEDESC) - input
Face-name description.

A structure that provides the characteristics of the required font. These characteristics are used to generate the compound face name.

length (LONG) - input
Length of *CompoundFaceName* buffer.

The maximum length of the compound face name returned (including the trailing zero of the string).

Specify zero to find out how large the *CompoundFaceName* buffer needs to be.

CompoundFaceName (PSZ) - output
Compound face name.

The compound face name of the font.

cbRetLength (ULONG) - returns
Length of the compound face name.

GPI_ERROR
Error occurred

> 0
Length of the compound face-name string (including the trailing zero). This is the length of the complete string; if it is greater than *length* , the string returned in *CompoundFaceName* is truncated.

GpiQueryFaceString - Remarks

This function generates a compound face name (for example, "Courier Bold Italic") from a family name (for example, "Courier").

The compound face name can be used on a GpiCreateLogFont function.

GpiQueryFaceString - Errors

Possible returns from WinGetLastError

PMERR_FONT_NOT_LOADED (0x202F)
An attempt was made to create a font that was not loaded.

PMERR_INV_FACENAME (0x210E)
An invalid font family name was passed to GpiQueryFaceString.

PMERR_INV_FACENAMEDESC (0x2115)
The font facename description is invalid.

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryFaceString - Example Code

In this example GpiQueryFaceString is used to generate a compound face name of "Courier Bold Italic" from the family name "Courier."

```
#define INCL_GPILCIDS           /* Font functions               */
#include <os2.h>

ULONG  cbRetLength;     /* length of compound face name         */
HPS   hps;              /* Presentation-space handle            */
char  pszFamilyName[13];/* Family name                          */
FACENAMEDESC  pfndFaceAttrs; /* Face-name description           */
LONG  lLength;          /* length of buffer                     */
char  pszCompoundFaceName[25];/* Compound face name             */

/* family name is 'Courier' */
strcpy(pszFamilyName,"Courier");

/* let the function determine the buffer length and return it */
lLength = 25L;
```

```
/* initialize face name description structure for bold weight
   class, normal width, and italics */
pfndFaceAttrs.usSize = sizeof(FACENAMEDESC);
                                  /* Length of structure  */
pfndFaceAttrs.usWeightClass = FWEIGHT_BOLD;  /* Weight class    */
pfndFaceAttrs.usWidthClass = FWIDTH_NORMAL;  /* Width class     */
pfndFaceAttrs.usReserved = 0;                /* Reserved        */
pfndFaceAttrs.flOptions = FTYPE_ITALIC;
                         /* Other characteristics of the font  */

cbRetLength = GpiQueryFaceString(hps, pszFamilyName,
                                 &pfndFaceAttrs, lLength,
                                 pszCompoundFaceName);
```

GpiQueryFaceString - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryFontAction

GpiQueryFontAction - Syntax

This function determines whether available fonts have been affected since the last time the function was called.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB      anchor;     /*  Anchor-block handle. */
ULONG    options;    /*  Options. */
ULONG    flActions;  /*  Actions indicator. */

flActions = GpiQueryFontAction(anchor, options);
```

GpiQueryFontAction Parameter - anchor

anchor (HAB) - input
Anchor-block handle.

GpiQueryFontAction Parameter - options

options (ULONG) - input
Options.

The following may be OR'ed together if required:

QFA_PUBLIC
Interested in any change of public fonts.
QFA_PRIVATE
Interested in any change of private fonts for the current process.

GpiQueryFontAction Return Value - flActions

flActions (ULONG) - returns
Actions indicator.

If no error occurs, QFA_PUBLIC and QFA_PRIVATE may be OR'ed together.

QFA_PUBLIC
A change of public fonts has occurred.
QFA_PRIVATE
A change of private fonts affecting the current process has occurred.
QFA_ERROR
Error occurred.

GpiQueryFontAction - Parameters

anchor (HAB) - input
Anchor-block handle.

options (ULONG) - input
Options.

The following may be OR'ed together if required:

QFA_PUBLIC
Interested in any change of public fonts.
QFA_PRIVATE
Interested in any change of private fonts for the current process.

flActions (ULONG) - returns
Actions indicator.

If no error occurs, QFA_PUBLIC and QFA_PRIVATE may be OR'ed together.

QFA_PUBLIC
A change of public fonts has occurred.
QFA_PRIVATE
A change of private fonts affecting the current process has occurred.

QFA_ERROR
Error occurred.

GpiQueryFontAction - Remarks

This function can be used by a font selection dialog to find out whether its database of font information is still valid.

The function returns that both public and private font changes have taken place the first time it is called on a given process.

GpiQueryFontAction - Errors

Possible returns from WinGetLastError

PMERR_INV_OR_INCOMPAT_OPTIONS (0x20A9)
An invalid or incompatible (with micro presentation space) options parameter was specified with GpiCreatePS or GpiSetPS.

GpiQueryFontAction - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Glossary

GpiQueryFontFileDescriptions

GpiQueryFontFileDescriptions - Syntax

This function determines whether a given file is a font resource file, and if so, returns the family and face names of the fonts that it contains.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB         hab;            /*  Anchor-block handle. */
PSZ         pszFilename;    /*  Fully qualified filename. */
PLONG       plCount;        /*  Pointer to the maximum number of family and face name pairs to be returned. */
PFFDESCS    affdescsNames;  /*  Array of font file descriptors. */
LONG        lRemFonts;      /*  Returns. */
```

```
lRemFonts = GpiQueryFontFileDescriptions(
             hab, pszFilename, plCount, affdescsNames);
```

GpiQueryFontFileDescriptions Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiQueryFontFileDescriptions Parameter - pszFilename

pszFilename (PSZ) - input
Fully qualified filename.

This is the name of the font resource. The filename extension is .FON.

GpiQueryFontFileDescriptions Parameter - plCount

plCount (PLONG) - in/out
Pointer to the maximum number of family and face name pairs to be returned.

A pointer to the number of pairs of descriptions that are actually returned in *affdescsNames* is returned in this variable.

GpiQueryFontFileDescriptions Parameter - affdescsNames

affdescsNames (PFFDESCS) - output
Array of font file descriptors.

An array of 2**plCount* consecutive 32-byte fields, in which the family and face names of each font, in turn, are returned alternately. For each pair, the family name is returned first.

GpiQueryFontFileDescriptions Return Value - lRemFonts

lRemFonts (LONG) - returns
Returns.

>=0

Number of fonts for which details were not returned.

GPI_ALTERROR
Error.

GpiQueryFontFileDescriptions - Parameters

hab (HAB) - input
Anchor-block handle.

pszFilename (PSZ) - input
Fully qualified filename.

This is the name of the font resource. The filename extension is .FON.

plCount (PLONG) - in/out
Pointer to the maximum number of family and face name pairs to be returned.

A pointer to the number of pairs of descriptions that are actually returned in *affdescsNames* is returned in this variable.

affdescsNames (PFFDESCS) - output
Array of font file descriptors.

An array of 2**plCount* consecutive 32-byte fields, in which the family and face names of each font, in turn, are returned alternately. For each pair, the family name is returned first.

lRemFonts (LONG) - returns
Returns.

>=0
Number of fonts for which details were not returned.

GPI_ALTERROR
Error.

GpiQueryFontFileDescriptions - Remarks

Details are returned for as many fonts as can be held in *affdescsNames* .

By inspecting the returned data, the application can tell whether a particular font resource file contains the fonts it requires, before loading it.

By specifying *plCount* as a pointer to 0, and then looking at the value returned by the font an application can determine how many fonts there are in the file, and then allocate the correct amount of buffer space for a subsequent call to obtain all of the names.

GpiQueryFontFileDescriptions - Errors

Possible returns from WinGetLastError

PMERR_INV_FONT_FILE_DATA (0x2073)
The font file specified with GpiLoadFonts, GpiLoadPublicFonts, GpiQueryFontFileDescriptions, or GpiQueryFullFontFileDescs contains invalid data.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

GpiQueryFontFileDescriptions - Example Code

This example uses the GpiQueryFontFileDescriptions to retrieve the typeface family and names for the fonts in the HELV.FON file. The function is called twice, once to determine the actual number of fonts in the file, and again to retrieve the descriptions.

```
#define INCL_GPILCIDS           /* Font functions               */
#define INCL_DOSMEMMGR          /* DOS Memory Manager Functions */
#include <os2.h>

HAB hab;                /* anchor-block handle                  */
PFFDESCS pffdescs;      /* array of font file descriptors       */
LONG cFonts = 0;        /* number of descriptions not returned  */

/* Retrieve a count of all fonts in the file. */

cFonts = GpiQueryFontFileDescriptions(hab, "C:\\HELV.FON", &cFonts,
                                      NULL);

/* Allocate space for the descriptions. */

ret = DosAllocMem((PPVOID *)&pffdescs,(ULONG)(cFonts*sizeof(FFDESCS)),
            PAG_COMMIT | PAG_READ | PAG_WRITE);
if (ret != 0) {
        DosBeep(100,100);
        err = WinGetLastError(hab);
        DosFreeMem(pffdescs);
        exit(-1);
}
/* Retrieve the descriptions. */

GpiQueryFontFileDescriptions(hab, "C:\\OS2\\DLL\\HELV.FON", &cFonts,
                             pffdescs);
```

GpiQueryFontFileDescriptions - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryFontMetrics

GpiQueryFontMetrics - Syntax

This function returns a record providing details of the font metrics for the logical font that is currently selected.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS             hps;             /*  Presentation-space handle. */
LONG            lMetricsLength;  /*  Length of metrics. */
PFONTMETRICS    pfmMetrics;      /*  Metrics of font. */
BOOL            rc;              /*  Success indicator. */

rc = GpiQueryFontMetrics(hps, lMetricsLength,
       pfmMetrics);
```

GpiQueryFontMetrics Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryFontMetrics Parameter - lMetricsLength

lMetricsLength (LONG) - input
Length of metrics.

It must be greater or equal to 0.

GpiQueryFontMetrics Parameter - pfmMetrics

pfmMetrics (PFONTMETRICS) - output
Metrics of font.

In this buffer are returned the font metrics of the logical font, identified by the current value of the character set attribute.

No more data than *lMetricsLength* is returned.

GpiQueryFontMetrics Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryFontMetrics - Parameters

hps (HPS) - input
Presentation-space handle.

lMetricsLength (LONG) - input
Length of metrics.

It must be greater or equal to 0.

pfmMetrics (PFONTMETRICS) - output
Metrics of font.

In this buffer are returned the font metrics of the logical font, identified by the current value of the character set attribute.

No more data than *lMetricsLength* is returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryFontMetrics - Remarks

All sizes are returned in world coordinates.

An application can determine if the font *szFacename* (as returned in *pfmMetrics*) has been truncated by checking the *fsType* field in *pfmMetrics* for the FM_TYPE_FACETRUNC indicator. If the face name has been truncated, this bit will be set, and the application can issue a WinQueryAtomName function, passing in the *FaceNameAtom* (as returned in *pfmMetrics*) to retrieve the full face name from the System Atom table.

GpiQueryFontMetrics - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW (0x2014)
An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiQueryFontMetrics - Example Code

This example uses the GpiQueryFontMetrics function to retrieve the font metrics for the current font.

```
#define INCL_GPILCIDS           /* Font functions               */
#include <os2.h>

HPS hps;                /* presentation space handle            */
FONTMETRICS fm;         /* metrics structure                    */

GpiQueryFontMetrics(hps, sizeof(FONTMETRICS), &fm);
```

GpiQueryFontMetrics - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryFonts

GpiQueryFonts - Syntax

This function returns a record providing details of the fonts that match the specified *pszFacename* .

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS             hps;            /*  Presentation-space handle. */
ULONG           flOptions;      /*  Enumeration options. */
PSZ             pszFacename;    /*  Face name of fonts. */
PLONG           plReqFonts;     /*  Count of fonts. */
LONG            lMetricsLength; /*  Length of metrics. */
PFONTMETRICS    afmMetrics;     /*  Metrics of font. */
LONG            lRemFonts;      /*  Count of fonts not returned. */

lRemFonts = GpiQueryFonts(hps, flOptions,
              pszFacename, plReqFonts, lMetricsLength,
              afmMetrics);
```

GpiQueryFonts Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryFonts Parameter - flOptions

flOptions (ULONG) - input
Enumeration options.

This controls which fonts are to be enumerated. If more than one of the following options are required, the values can be ORed together.

QF_PUBLIC
: Enumerate public fonts.

QF_PRIVATE
: Enumerate private fonts.

QF_NO_DEVICE
: Device fonts are not reported.

QF_NO_GENERIC
: Generic fonts are not reported.

GpiQueryFonts Parameter - pszFacename

pszFacename (PSZ) - input
Face name of fonts.

If the pointer to *pszFacename* is NULL, all available fonts are queried, regardless of their face names.

GpiQueryFonts Parameter - plReqFonts

plReqFonts (PLONG) - in/out
Count of fonts.

The number of fonts for which the application requires the metrics. This variable returns the number of fonts returned.

GpiQueryFonts Parameter - lMetricsLength

lMetricsLength (LONG) - input

Length of metrics.

The length of each metrics record to be returned. The *afmMetrics* data area must be *plReqFonts* multiplied by *lMetricsLength* long.

GpiQueryFonts Parameter - afmMetrics

afmMetrics (PFONTMETRICS) - output
Metrics of font.

In this structure are returned the font metrics of up to *plReqFonts* matching fonts. The format for each record is as defined for GpiQueryFontMetrics, except that the *usCodePage* field has no meaning in this context, and is indeterminate. For each font, no more data than *lMetricsLength* is returned.

GpiQueryFonts Return Value - lRemFonts

lRemFonts (LONG) - returns
Count of fonts not returned.

>=0
Count of fonts not returned
GPI_ALTERROR
Error.

GpiQueryFonts - Parameters

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Enumeration options.

This controls which fonts are to be enumerated. If more than one of the following options are required, the values can be ORed together.

QF_PUBLIC
Enumerate public fonts.
QF_PRIVATE
Enumerate private fonts.
QF_NO_DEVICE
Device fonts are not reported.
QF_NO_GENERIC
Generic fonts are not reported.

pszFacename (PSZ) - input
Face name of fonts.

If the pointer to *pszFacename* is NULL, all available fonts are queried, regardless of their face names.

plReqFonts (PLONG) - in/out
Count of fonts.

The number of fonts for which the application requires the metrics. This variable returns the number of fonts returned.

lMetricsLength (LONG) - input
Length of metrics.

The length of each metrics record to be returned. The *afmMetrics* data area must be *plReqFonts* multiplied by *lMetricsLength* long.

afmMetrics (PFONTMETRICS) - output
Metrics of font.

In this structure are returned the font metrics of up to *plReqFonts* matching fonts. The format for each record is as defined for GpiQueryFontMetrics, except that the *usCodePage* field has no meaning in this context, and is indeterminate. For each font, no more data than *lMetricsLength* is returned.

lRemFonts (LONG) - returns
Count of fonts not returned.

>=0
Count of fonts not returned
GPI_ALTERROR
Error.

GpiQueryFonts - Remarks

Font metrics are returned for as many matching fonts as can be held in *afmMetrics* .

By inspecting the returned data, the application can choose which of the available fonts is most appropriate for its requirements. If necessary, it can force selection of a particular font, by specifying its match (as returned in *afmMetrics*) in the *pfatAttrs* structure for GpiCreateLogFont, however, this is only valid for a particular device/device driver combination on a single machine. This method should be avoided as a method for selecting a font.

An application can determine if the font *szFacename* (as returned in *afmMetrics*) has been truncated by checking the *fsType* field in *afmMetrics* for the FM_TYPE_FACETRUNC indicator. If the face name has been truncated, this bit will be set, and the application can issue a WinQueryAtomName function, passing in the *FaceNameAtom* (as returned in *afmMetrics*) to retrieve the full face name from the System Atom table.

By specifying *plReqFonts* as 0, and then looking at the value returned in *lRemFonts* , an application can determine how many fonts there are that match the *pszFacename* .

All sizes are returned in world coordinates.

GpiQueryFonts - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW (0x2014)
An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiQueryFonts - Example Code

This example uses the GpiQueryFonts function to retrieve the font metrics for all private fonts having the "Helv" typeface name. The function is called twice, first to determine the number of fonts available, and then again to retrieve the font metrics for all the fonts.

```
#define INCL_GPILCIDS           /* Font functions               */
#define INCL_DOSMEMMGR          /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;                /* presentation space handle            */
LONG cFonts;            /* fonts not returned                   */
LONG lTemp = 0L;        /* font count                           */
PFONTMETRICS pfm;       /* metrics structure                    */

/* Determine the number of fonts. */

cFonts = GpiQueryFonts(hps, QF_PRIVATE, "Helv", &lTemp,
    (LONG) sizeof(FONTMETRICS), NULL);

/* Allocate space for the font metrics. */

DosAllocMem((VOID *)pfm,(ULONG)(cFonts*sizeof(FONTMETRICS)),
            PAG_COMMIT | PAG_READ | PAG_WRITE);

/* Retrieve the font metrics. */

cFonts = GpiQueryFonts(hps, QF_PRIVATE, "Helv", &cFonts,
    (LONG) sizeof(FONTMETRICS), pfm);
```

GpiQueryFonts - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryFullFontFileDescs

GpiQueryFullFontFileDescs - Syntax

This function determines whether a given file is a font resource file, and if so, returns the family and face names of the fonts that it contains.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB      hab;               /*  Anchor-block handle. */
```

```
PSZ      pszFilename;        /*  Fully qualified filename. */
PLONG    plCount;            /*  Maximum number of family and face name pairs to be returned. */
PVOID    pNames;             /*  Font file descriptors. */
PLONG    plNamesBuffLength;  /*  Length, in bytes, of the pNames data buffer. */
LONG     lRemFonts;          /*  Returns. */

lRemFonts = GpiQueryFullFontFileDescs(hab,
              pszFilename, plCount, pNames,
              plNamesBuffLength);
```

GpiQueryFullFontFileDescs Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiQueryFullFontFileDescs Parameter - pszFilename

pszFilename (PSZ) - input
Fully qualified filename.

This is the name of the font resource. The filename extension is .FON.

GpiQueryFullFontFileDescs Parameter - plCount

plCount (PLONG) - in/out
Maximum number of family and face name pairs to be returned.

The number of pairs of descriptions that are actually returned in *pNames* is returned in this variable.

GpiQueryFullFontFileDescs Parameter - pNames

pNames (PVOID) - output
Font file descriptors.

A buffer in which the font file family and face name pairs are returned. They are each returned in a FFDESCS2 structure, with successive structures packed end to end.

GpiQueryFullFontFileDescs Parameter - plNamesBuffLength

plNamesBuffLength (PLONG) - in/out
Length, in bytes, of the *pNames* data buffer.

On return, this is set to the actual length needed to hold all of the family names and face names in the file.

GpiQueryFullFontFileDescs Return Value - lRemFonts

lRemFonts (LONG) - returns
Returns.

>=0
Number of fonts for which details were not returned
GPI_ALTERROR
Error.

GpiQueryFullFontFileDescs - Parameters

hab (HAB) - input
Anchor-block handle.

pszFilename (PSZ) - input
Fully qualified filename.

This is the name of the font resource. The filename extension is .FON.

plCount (PLONG) - in/out
Maximum number of family and face name pairs to be returned.

The number of pairs of descriptions that are actually returned in *pNames* is returned in this variable.

pNames (PVOID) - output
Font file descriptors.

A buffer in which the font file family and face name pairs are returned. They are each returned in a FFDESCS2 structure, with successive structures packed end to end.

plNamesBuffLength (PLONG) - in/out
Length, in bytes, of the *pNames* data buffer.

On return, this is set to the actual length needed to hold all of the family names and face names in the file.

lRemFonts (LONG) - returns
Returns.

>=0
Number of fonts for which details were not returned
GPI_ALTERROR
Error.

GpiQueryFullFontFileDescs - Remarks

Details are returned for as many fonts as can be held in *pNames* .

By inspecting the returned data, the application can tell whether a particular font resource file contains the fonts it requires, before loading it.

By specifying *pNames* as NULL, and then looking at the value returned in *plNamesBuffLength* , an application can determine the length of the buffer needed to hold all of the font names.

Support for this function is device dependent.

GpiQueryFullFontFileDescs - Errors

Possible returns from WinGetLastError

PMERR_INV_FONT_FILE_DATA (0x2073)
: The font file specified with GpiLoadFonts, GpiLoadPublicFonts, GpiQueryFontFileDescriptions, or GpiQueryFullFontFileDescs contains invalid data.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

GpiQueryFullFontFileDescs - Example Code

This example uses the GpiQueryFullFontFileDescs to retrieve the typeface family and names for the fonts in the HELV.FON file. The function is called twice, once to determine the actual number of fonts in the file, and again to retrieve the descriptions.

```
HAB  hab;
PFFDESCS pffdescs;
LONG cFonts = 0;
LONG lBuflen = 0;

/* Retrieve a count of all fonts in the file. */

cFonts = GpiQueryFontFileDescriptions(hab, "C:\\HELV.FON",
                                      &cFonts, NULL, &lBuflen);

/* Allocate space for the descriptions. */

DosAllocMem((VOID *)pffdescs,(ULONG)(cFonts*sizeof(FFDESCS)),
            PAG_COMMIT | PAG_READ | PAG_WRITE);

/* Retrieve the descriptions. */

GpiQueryFullFontFileDescs(hab, "C:\\HELV.FON", &cFonts,
                          pffdescs, &lBuflen);
```

GpiQueryFullFontFileDescs - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks

Example Code
Glossary

GpiQueryGraphicsField

GpiQueryGraphicsField - Syntax

This function returns the bottom-left and top-right corners of the graphics field in presentation page units, as set by the GpiSetGraphicsField function.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
PRECTL    prclField;  /*  Graphics field. */
BOOL      rc;         /*  Success indicator. */

rc = GpiQueryGraphicsField(hps, prclField);
```

GpiQueryGraphicsField Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryGraphicsField Parameter - prclField

prclField (PRECTL) - output
Graphics field.

GpiQueryGraphicsField Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiQueryGraphicsField - Parameters

hps (HPS) - input
Presentation-space handle.

prclField (PRECTL) - output
Graphics field.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryGraphicsField - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryGraphicsField - Example Code

This example uses GpiQueryGraphicsField to return the bottom-left and top-right corners of the graphics field in presentation page units, as set by the GpiSetGraphicsField call, and then assigns the x coordinate of the lower left hand field corner to a variable.

```
#define INCL_GPITRANSFORMS      /* Transform functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
RECTL prclField;        /* graphics field                       */
LONG   lLwrLftxCoord;   /* lower left x coordinate of field     */

fSuccess = GpiQueryGraphicsField(hps, &prclField);

/* if successful, assign lower left x coordinate of graphics field */
if (fSuccess == TRUE)
   lLwrLftxCoord = prclField.xLeft;
```

GpiQueryGraphicsField - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryInitialSegmentAttrs

GpiQueryInitialSegmentAttrs - Syntax

This function returns the current value of a particular initial segment attribute.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;         /*  Presentation-space handle. */
LONG    lAttribute;  /*  Attribute to be queried. */
LONG    lValue;      /*  Current initial attribute value. */

lValue = GpiQueryInitialSegmentAttrs(hps,
           lAttribute);
```

GpiQueryInitialSegmentAttrs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryInitialSegmentAttrs Parameter - lAttribute

lAttribute (LONG) - input
Attribute to be queried.

Identifies the attribute to be returned by this function:

ATTR_DETECTABLE
Detectability
ATTR_VISIBLE
Visibility
ATTR_CHAINED

Chained
ATTR_DYNAMIC
Dynamic
ATTR_FASTCHAIN
Fast chaining
ATTR_PROP_DETECTABLE
Propagate detectability
ATTR_PROP_VISIBLE
Propagate visibility.

GpiQueryInitialSegmentAttrs Return Value - lValue

lValue (LONG) - returns
Current initial attribute value.

ATTR_ON
On/yes
ATTR_OFF
Off/no
ATTR_ERROR
Error.

GpiQueryInitialSegmentAttrs - Parameters

hps (HPS) - input
Presentation-space handle.

lAttribute (LONG) - input
Attribute to be queried.

Identifies the attribute to be returned by this function:

ATTR_DETECTABLE
Detectability
ATTR_VISIBLE
Visibility
ATTR_CHAINED
Chained
ATTR_DYNAMIC
Dynamic
ATTR_FASTCHAIN
Fast chaining
ATTR_PROP_DETECTABLE
Propagate detectability
ATTR_PROP_VISIBLE
Propagate visibility.

lValue (LONG) - returns
Current initial attribute value.

ATTR_ON
On/yes
ATTR_OFF
Off/no
ATTR_ERROR
Error.

GpiQueryInitialSegmentAttrs - Remarks

Initial segment attributes are modal settings used to determine the initial attributes of new segments as those new segments are created; see GpiSetInitialSegmentAttrs.

GpiQueryInitialSegmentAttrs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_ATTR (0x20C5)
: An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQueryInitialSegmentAttrs - Example Code

This example uses GpiQueryInitialSegmentAttrs to queries the current state of the dynamic segment attribute.

```
#define INCL_GPISEGMENTS        /* Segment functions            */
#include <os2.h>

LONG     lValue;        /* current element pointer             */
HPS    hps;             /* Presentation-space handle           */
LONG     lAttribute;    /* attribute to query                  */

/* query the dynamic attribute */
lAttribute = ATTR_DYNAMIC;

lValue = GpiQueryInitialSegmentAttrs(hps, lAttribute);
```

GpiQueryInitialSegmentAttrs - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryKerningPairs

GpiQueryKerningPairs - Syntax

This function returns kerning pair information for the logical font identified by the current value of the character set attribute.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS             hps;         /*  Presentation-space handle. */
LONG            lCount;      /*  Number of elements in akrnprData. */
PKERNINGPAIRS   akrnprData;  /*  Kerning pairs. */
LONG            lReturned;   /*  Number returned and error indicators. */

lReturned = GpiQueryKerningPairs(hps, lCount,
             akrnprData);
```

GpiQueryKerningPairs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryKerningPairs Parameter - lCount

lCount (LONG) - input
Number of elements in *akrnprData* .

GpiQueryKerningPairs Parameter - akrnprData

akrnprData (PKERNINGPAIRS) - output
Kerning pairs.

An array of *lCount* kerning pairs in which information is returned. No more than *lCount* records are returned.

GpiQueryKerningPairs Return Value - lReturned

lReturned (LONG) - returns
Number returned and error indicators.

>=0
Number of kerning pairs returned
GPI_ALTERROR
Error.

GpiQueryKerningPairs - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of elements in *akrnprData* .

akrnprData (PKERNINGPAIRS) - output
Kerning pairs.

An array of *lCount* kerning pairs in which information is returned. No more than *lCount* records are returned.

lReturned (LONG) - returns
Number returned and error indicators.

>=0
Number of kerning pairs returned
GPI_ALTERROR
Error.

GpiQueryKerningPairs - Remarks

The number of kerned pairs is a field in the font metrics.

GpiQueryKerningPairs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW (0x2014)
An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or

too large.

GpiQueryKerningPairs - Example Code

This example uses the GpiQueryKerningPairs function to retrieve the kerning information for the current font.

```
#define INCL_GPILCIDS           /* Font functions               */
#define INCL_DOSMEMMGR          /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;                /* presentation space handle            */
FONTMETRICS fm;         /* metrics structure                    */
PKERNINGPAIRS akrnpr;   /* kerning pairs array                  */

GpiQueryFontMetrics(hps, (LONG) sizeof(FONTMETRICS), &fm);

DosAllocMem((VOID *)akrnpr,
            (ULONG)(fm.sKerningPairs * sizeof(KERNINGPAIRS)),
            PAG_COMMIT | PAG_READ | PAG_WRITE);

GpiQueryKerningPairs(hps, (LONG) fm.sKerningPairs, akrnpr);
```

GpiQueryKerningPairs - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryLineEnd

GpiQueryLineEnd - Syntax

This function returns the current line-end attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lLineEnd;  /*  Line end. */

lLineEnd = GpiQueryLineEnd(hps);
```

GpiQueryLineEnd Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryLineEnd Return Value - lLineEnd

lLineEnd (LONG) - returns
Line end.

LINEEND_DEFAULT
(0L) Default
LINEEND_FLAT
(1L) Flat
LINEEND_SQUARE
(2L) Square
LINEEND_ROUND
(3L) Round
LINEEND_ERROR
(-1L) Error.

GpiQueryLineEnd - Parameters

hps (HPS) - input
Presentation-space handle.

lLineEnd (LONG) - returns
Line end.

LINEEND_DEFAULT
(0L) Default
LINEEND_FLAT
(1L) Flat
LINEEND_SQUARE
(2L) Square
LINEEND_ROUND
(3L) Round
LINEEND_ERROR
(-1L) Error.

GpiQueryLineEnd - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryLineEnd - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
: An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryLineEnd - Example Code

This example uses GpiQueryLineEnd to return the current line-end attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
LONG  lLineEnd;         /* Line end                            */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   lLineEnd = GpiQueryLineEnd(hps);
```

GpiQueryLineEnd - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryLineJoin

GpiQueryLineJoin - Syntax

This function returns the current line-join attribute, as set by the GpiSetLineJoin function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;        /*  Presentation-space handle. */
LONG    lLineJoin;  /*  Line join. */

lLineJoin = GpiQueryLineJoin(hps);
```

GpiQueryLineJoin Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryLineJoin Return Value - lLineJoin

lLineJoin (LONG) - returns
Line join.

LINEJOIN_DEFAULT
(0L) Default
LINEJOIN_BEVEL
(1L) Bevel
LINEJOIN_ROUND
(2L) Round
LINEJOIN_MITRE
(3L) Miter
LINEJOIN_ERROR
(-1L)Error.

GpiQueryLineJoin - Parameters

hps (HPS) - input
Presentation-space handle.

lLineJoin (LONG) - returns
Line join.

LINEJOIN_DEFAULT
(0L) Default
LINEJOIN_BEVEL
(1L) Bevel

LINEJOIN_ROUND
(2L) Round
LINEJOIN_MITRE
(3L) Miter
LINEJOIN_ERROR
(-1L)Error.

GpiQueryLineJoin - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryLineJoin - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryLineJoin - Example Code

This example uses GpiQueryLineJoin to return the current line-join attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
LONG  lLineJoin;        /* Line join                           */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   lLineJoin = GpiQueryLineJoin(hps);
```

GpiQueryLineJoin - Topics

Select an item:
Syntax
Parameters

Returns
Errors
Remarks
Example Code
Glossary

GpiQueryLineType

GpiQueryLineType - Syntax

This function returns the current cosmetic line-type attribute, as set by the GpiSetLineType function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;        /*  Presentation-space handle. */
LONG    lLineType;  /*  Line type. */

lLineType = GpiQueryLineType(hps);
```

GpiQueryLineType Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryLineType Return Value - lLineType

lLineType (LONG) - returns
Line type.

LINETYPE_DEFAULT
(0L) Default
LINETYPE_DOT
(1L) Dotted line
LINETYPE_SHORTDASH
(2L) Short-dashed line
LINETYPE_DASHDOT
(3L) Dash-dotted line
LINETYPE_DOUBLEDOT
(4L) Double-dotted line
LINETYPE_LONGDASH
(5L) Long-dashed line
LINETYPE_DASHDOUBLE DOT

(6L) Dash-double dotted line
LINETYPE_SOLID
(7L) Solid line
LINETYPE_ALTERNATE
(9L) Alternate pale on
LINETYPE_INVISIBLE
(8L) Invisible line
LINETYPE_ERROR
(-1L) Error.

GpiQueryLineType - Parameters

hps (HPS) - input
Presentation-space handle.

lLineType (LONG) - returns
Line type.

LINETYPE_DEFAULT
(0L) Default
LINETYPE_DOT
(1L) Dotted line
LINETYPE_SHORTDASH
(2L) Short-dashed line
LINETYPE_DASHDOT
(3L) Dash-dotted line
LINETYPE_DOUBLEDOT
(4L) Double-dotted line
LINETYPE_LONGDASH
(5L) Long-dashed line
LINETYPE_DASHDOUBLE DOT
(6L) Dash-double dotted line
LINETYPE_SOLID
(7L) Solid line
LINETYPE_ALTERNATE
(9L) Alternate pale on
LINETYPE_INVISIBLE
(8L) Invisible line
LINETYPE_ERROR
(-1L) Error.

GpiQueryLineType - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryLineType - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryLineType - Example Code

This example uses GpiQueryLineType to return the current cosmetic line-type attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
LONG  lLineType;        /* Line type                           */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   lLineType = GpiQueryLineType(hps);
```

GpiQueryLineType - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryLineWidth

GpiQueryLineWidth - Syntax

This function returns the current value of the cosmetic line-width attribute, as set by the GpiSetLineWidth function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;          /*  Presentation-space handle. */
FIXED    fxLineWidth;  /*  Line width. */
```

```
fxLineWidth = GpiQueryLineWidth(hps);
```

GpiQueryLineWidth Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryLineWidth Return Value - fxLineWidth

fxLineWidth (FIXED) - returns
Line width.

LINEWIDTH_DEFAULT
(0L) Default
LINEWIDTH_NORMAL
(1L) Normal width (1.0)
LINEWIDTH_THICK
(2L) Thick line
LINEWIDTH_ERROR
(-1L) Error.

GpiQueryLineWidth - Parameters

hps (HPS) - input
Presentation-space handle.

fxLineWidth (FIXED) - returns
Line width.

LINEWIDTH_DEFAULT
(0L) Default
LINEWIDTH_NORMAL
(1L) Normal width (1.0)
LINEWIDTH_THICK
(2L) Thick line
LINEWIDTH_ERROR
(-1L) Error.

GpiQueryLineWidth - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryLineWidth - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryLineWidth - Example Code

This example uses GpiQueryLineWidth to return the current cosmetic line-width attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
FIXED fxLineWidth;      /* Line width                          */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   fxLineWidth = GpiQueryLineWidth(hps);
```

GpiQueryLineWidth - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryLineWidthGeom

GpiQueryLineWidthGeom - Syntax

This function returns the current geometric line-width attribute, as set by the GpiSetLineWidthGeom function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;         /*  Presentation-space handle. */
LONG    lLineWidth;  /*  Geometric line width. */

lLineWidth = GpiQueryLineWidthGeom(hps);
```

GpiQueryLineWidthGeom Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryLineWidthGeom Return Value - lLineWidth

lLineWidth (LONG) - returns
Geometric line width.

If the geometric line width is currently set to the default, zero is returned.

>=0
Geometric line width
LINEWIDTHGEOM_ERROR
Error.

GpiQueryLineWidthGeom - Parameters

hps (HPS) - input
Presentation-space handle.

lLineWidth (LONG) - returns
Geometric line width.

If the geometric line width is currently set to the default, zero is returned.

>=0
Geometric line width
LINEWIDTHGEOM_ERROR
Error.

GpiQueryLineWidthGeom - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryLineWidthGeom - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryLineWidthGeom - Example Code

This example uses GpiQueryLineWidthGeom to return the current geometric line-width attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
LONG  lLineWidth;       /* geometric line width                */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   lLineWidth = GpiQueryLineWidthGeom(hps);
```

GpiQueryLineWidthGeom - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryLogColorTable

GpiQueryLogColorTable - Syntax

This function returns the logical color table.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
ULONG    flOptions;  /*  Specifies options. */
LONG     lStart;     /*  Starting index for which data is to be returned. */
LONG     lCount;     /*  Count of elements. */
PLONG    alArray;    /*  Array in which the information is returned. */
LONG     lRetCount;  /*  Number of elements returned and error indicators. */

lRetCount = GpiQueryLogColorTable(hps, flOptions,
              lStart, lCount, alArray);
```

GpiQueryLogColorTable Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryLogColorTable Parameter - flOptions

flOptions (ULONG) - input
Specifies options.

LCOLOPT_INDEX
B'1' The index is to be returned for each RGB value.

Other flags are reserved and must be B'0'.

GpiQueryLogColorTable Parameter - lStart

lStart (LONG) - input
Starting index for which data is to be returned.

This must be greater than or equal to zero.

GpiQueryLogColorTable Parameter - lCount

lCount (LONG) - input
Count of elements.

Number of elements available in *alArray* . It must be greater or equal to 0.

GpiQueryLogColorTable Parameter - alArray

alArray (PLONG) - output
Array in which the information is returned.

If the LCOLOPT_INDEX flag is B'0', it is an array of RGB values (each value is as defined for GpiCreateLogColorTable), starting with the specified index, and ending either when there are no further loaded entries in the table, or when *alArray* has been exhausted. If the logical color table is not loaded with a contiguous set of indexes, QLCT_NOTLOADED is returned as the RGB value for any index values, outside the default range, that have not been explicitly loaded.

If the LCOLOPT_INDEX flag is B'1', it is an array of alternating color indexes and RGB values, in the order index1, RGB value1, index2, RGB value2,... An even number of elements is always returned. If the logical color table is not loaded with a contiguous set of indexes, any index values that are not loaded are skipped.

GpiQueryLogColorTable Return Value - lRetCount

lRetCount (LONG) - returns
Number of elements returned and error indicators.

QLCT_RGB
Table in RGB mode, no elements returned
>0
Number of elements returned
QLCT_ERROR
Error.

GpiQueryLogColorTable - Parameters

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Specifies options.

LCOLOPT_INDEX
B'1' The index is to be returned for each RGB value.

Other flags are reserved and must be B'0'.

lStart (LONG) - input
Starting index for which data is to be returned.

This must be greater than or equal to zero.

lCount (LONG) - input
Count of elements.

Number of elements available in *alArray*. It must be greater or equal to 0.

alArray (PLONG) - output
Array in which the information is returned.

If the LCOLOPT_INDEX flag is B'0', it is an array of RGB values (each value is as defined for GpiCreateLogColorTable), starting with the specified index, and ending either when there are no further loaded entries in the table, or when *alArray* has been exhausted. If the logical color table is not loaded with a contiguous set of indexes, QLCT_NOTLOADED is returned as the RGB value for any index values, outside the default range, that have not been explicitly loaded.

If the LCOLOPT_INDEX flag is B'1', it is an array of alternating color indexes and RGB values, in the order index1, RGB value1, index2, RGB value2,... An even number of elements is always returned. If the logical color table is not loaded with a contiguous set of indexes, any index values that are not loaded are skipped.

lRetCount (LONG) - returns
Number of elements returned and error indicators.

QLCT_RGB
: Table in RGB mode, no elements returned

>0
: Number of elements returned

QLCT_ERROR
: Error.

GpiQueryLogColorTable - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COLOR_OPTIONS (0x2057)
An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_COLOR_START_INDEX (0x2058)
An invalid starting index parameter was specified with a logical color table or color query function.

PMERR_PALETTE_SELECTED (0x210F)
Color palette operations cannot be performed on a presentation space while a palette is selected.

GpiQueryLogColorTable - Example Code

This example uses the GpiQueryLogColorTable function to retrieve all the entries in the current logical color table.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
```

```
#define INCL_DOSMEMMGR          /* DOS Memory Manager Functions */
#define INCL_DEV                /* Device Function definitions  */
#include <os2.h>

HPS hps;                /* presentation space handle            */
LONG cColors;           /* number of colors                     */
PLONG alColor;          /* color table array                    */

/* Find out how many colors are in the color table. */

DevQueryCaps(GpiQueryDevice(hps), CAPS_COLORS, 1L, &cColors);

/* Allocate space for the color values and indexes. */

DosAllocMem((VOID *)alColor,(ULONG)cColors*2,
            PAG_COMMIT | PAG_READ | PAG_WRITE);

/* Retrieve the values. */

GpiQueryLogColorTable(hps,   /* presentation space              */
    LCOLOPT_INDEX,           /* retrieve indexes and RGB values */
    0L,                      /* start with first entry          */
    cColors * 2,             /* copy 2 values for each entry    */
    alColor);                /* array to receive values         */
```

GpiQueryLogColorTable - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryLogicalFont

GpiQueryLogicalFont - Syntax

This function returns the description of a logical font. See GpiCreateLogFont.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        ps;      /*  Presentation-space handle. */
LONG       lcid;    /*  Local identifier. */
PSTR8      name;    /*  Logical font name. */
PFATTRS    attrs;   /*  Attributes of font. */
LONG       length;  /*  Length of attrs buffer. */
BOOL       rc;      /*  Success indicator. */

rc = GpiQueryLogicalFont(ps, lcid, name, attrs,
       length);
```

GpiQueryLogicalFont Parameter - ps

ps (HPS) - input
Presentation-space handle.

GpiQueryLogicalFont Parameter - lcid

lcid (LONG) - input
Local identifier.

Logical font local identifier, in the range 0 through 254. Specify 0 to query the default font.

GpiQueryLogicalFont Parameter - name

name (PSTR8) - output
Logical font name.

An 8-character name for the logical font.

GpiQueryLogicalFont Parameter - attrs

attrs (PFATTRS) - output
Attributes of font.

GpiQueryLogicalFont Parameter - length

length (LONG) - input
Length of *attrs* buffer.

The maximum length of font attribute data to be returned.

GpiQueryLogicalFont Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryLogicalFont - Parameters

ps (HPS) - input
Presentation-space handle.

lcid (LONG) - input
Local identifier.

Logical font local identifier, in the range 0 through 254. Specify 0 to query the default font.

name (PSTR8) - output
Logical font name.

An 8-character name for the logical font.

attrs (PFATTRS) - output
Attributes of font.

length (LONG) - input
Length of *attrs* buffer.

The maximum length of font attribute data to be returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryLogicalFont - Remarks

If the specified local identifier is in use to tag a bit map (see GpiSetBitmapId), an error is raised.

GpiQueryLogicalFont - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SETID (0x20CA)
An invalid setid parameter was specified.

PMERR_SETID_IN_USE (0x2102)
An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

GpiQueryLogicalFont - Example Code

This example uses GpiQueryLogicalFont to return the description of the default logical font and if the query succeeds, assigns the font code page to a variable.

```
#define INCL_GPILCIDS           /* Font functions               */
#include <os2.h>

BOOL     fSuccess;        /* success indicator                  */
HPS      hps;             /* Presentation-space handle          */
LONG     lLcid;           /* local identifier                   */
STR8     Name;            /* 8 character logical font name      */
FATTRS   fatAttrs;        /* Attributes of font                 */
LONG     lAttrsLength;    /* length of buffer                   */
USHORT   usFontCodePage;  /* font code page                     */

/* query the default font */
lLcid = 0L;

/* return all information */
lAttrsLength = sizeof(FATTRS);

fSuccess = GpiQueryLogicalFont(hps, lLcid, &Name, &fatAttrs,
                               lAttrsLength);

/* if successful, assign value of font code page */
if (fSuccess == TRUE)
   usFontCodePage = fatAttrs.usCodePage;
```

GpiQueryLogicalFont - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryMarker

GpiQueryMarker - Syntax

This function returns the current value of the marker symbol attribute, as set by the GpiSetMarker function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;      /*  Presentation-space handle. */
LONG    lSymbol;  /*  Marker symbol. */

lSymbol = GpiQueryMarker(hps);
```

GpiQueryMarker Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryMarker Return Value - lSymbol

lSymbol (LONG) - returns
Marker symbol.

MARKSYM_DEFAULT
(0L) Default
MARKSYM_CROSS
(1L) Cross symbol
MARKSYM_PLUS
(2L) Plus symbol
MARKSYM_DIAMOND
(3L) Diamond symbol
MARKSYM_SQUARE
(4L) Square symbol
MARKSYM_SIXPOINTSTAR
(5L) Six-point star symbol
MARKSYM_EIGHTPOINTSTAR
(6L) Eight-point star symbol
MARKSYM_SOLIDDIAMOND
(7L) Solid diamond symbol
MARKSYM_SOLIDSQUARE
(8L) Solid square symbol
MARKERSYM_DOT
(9L) Dot symbol
MARKSYM_SMALLCIRLCE
(10L) Small Circle symbol
MARKSYM_BLANK
(64L) Blank symbol
MARKSYM_ERROR
(-1L) Error

GpiQueryMarker - Parameters

hps (HPS) - input
Presentation-space handle.

lSymbol (LONG) - returns
Marker symbol.

MARKSYM_DEFAULT
(0L) Default
MARKSYM_CROSS
(1L) Cross symbol
MARKSYM_PLUS
(2L) Plus symbol
MARKSYM_DIAMOND
(3L) Diamond symbol
MARKSYM_SQUARE
(4L) Square symbol
MARKSYM_SIXPOINTSTAR
(5L) Six-point star symbol
MARKSYM_EIGHTPOINTSTAR
(6L) Eight-point star symbol
MARKSYM_SOLIDDIAMOND
(7L) Solid diamond symbol
MARKSYM_SOLIDSQUARE
(8L) Solid square symbol
MARKERSYM_DOT
(9L) Dot symbol
MARKSYM_SMALLCIRLCE
(10L) Small Circle symbol
MARKSYM_BLANK
(64L) Blank symbol
MARKSYM_ERROR
(-1L) Error

GpiQueryMarker - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryMarker - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryMarker - Example Code

This example uses GpiQueryMarker to return the current marker symbol attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
LONG  lSymbol;          /* marker symbol                       */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   lSymbol = GpiQueryMarker(hps);
```

GpiQueryMarker - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryMarkerBox

GpiQueryMarkerBox - Syntax

This function returns the current value of the marker-box attribute, as set by the GpiSetMarkerBox function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;         /*  Presentation-space handle. */
PSIZEF    psizfxSize;  /*  Size of marker box. */
BOOL      rc;          /*  Success indicator. */

rc = GpiQueryMarkerBox(hps, psizfxSize);
```

GpiQueryMarkerBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryMarkerBox Parameter - psizfxSize

psizfxSize (PSIZEF) - output
Size of marker box.

The size of the marker box is in world coordinates.

If the marker box is currently set to the default, the default size is returned.

GpiQueryMarkerBox Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryMarkerBox - Parameters

hps (HPS) - input
Presentation-space handle.

psizfxSize (PSIZEF) - output
Size of marker box.

The size of the marker box is in world coordinates.

If the marker box is currently set to the default, the default size is returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryMarkerBox - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryMarkerBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryMarkerBox - Example Code

This example uses GpiQueryMarkerBox to return the current marker-box attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
SIZEF  psizfxSize;      /* size of marker-box                   */
FIXED  lWidth;          /* marker-box width                     */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   fSuccess = GpiQueryMarkerBox(hps, &psizfxSize);

/* if successful, assign value of marker-box width */
if (fSuccess == TRUE)
   lWidth = psizfxSize.cx;
```

GpiQueryMarkerBox - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryMarkerSet

GpiQueryMarkerSet - Syntax

This function returns the current value of the marker-set attribute, as set by the GpiSetMarkerSet function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
LONG    lSet;  /*  Marker-set local identifier. */

lSet = GpiQueryMarkerSet(hps);
```

GpiQueryMarkerSet Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryMarkerSet Return Value - lSet

lSet (LONG) - returns
Marker-set local identifier.

LCID_DEFAULT
Default
>0
Marker-set local identifier
LCID_ERROR
Error.

GpiQueryMarkerSet - Parameters

hps (HPS) - input
Presentation-space handle.

lSet (LONG) - returns
Marker-set local identifier.

LCID_DEFAULT
: Default

>0
: Marker-set local identifier

LCID_ERROR
: Error.

GpiQueryMarkerSet - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryMarkerSet - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
: An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryMarkerSet - Example Code

This example uses GpiQueryMarkerSet to return the current marker-set attribute after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
LONG  lSet;             /* marker-set local identifier         */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   lSet = GpiQueryMarkerSet(hps);
```

GpiQueryMarkerSet - Topics

Select an item:
Syntax

Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryMetaFileBits

GpiQueryMetaFileBits - Syntax

This function transfers a metafile to application storage.

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HMF      hmf;      /*  Memory-metafile handle. */
LONG     lOffset;  /*  Byte offset. */
LONG     lLength;  /*  Length in bytes of the metafile data to copy. */
PBYTE    pbData;   /*  Metafile data. */
BOOL     rc;       /*  Success indicator. */

rc = GpiQueryMetaFileBits(hmf, lOffset, lLength,
       pbData);
```

GpiQueryMetaFileBits Parameter - hmf

hmf (HMF) - input
Memory-metafile handle.

GpiQueryMetaFileBits Parameter - lOffset

lOffset (LONG) - input
Byte offset.

Offset into the metafile data from which the transfer must start. This is useful in instances where the metafile data is too long to fit into a single application buffer. It must be greater or equal to 0.

GpiQueryMetaFileBits Parameter - lLength

lLength (LONG) - input
Length in bytes of the metafile data to copy.

It must be greater or equal to 0.

GpiQueryMetaFileBits Parameter - pbData

pbData (PBYTE) - output
Metafile data.

Address in application storage into which the metafile data is copied.

GpiQueryMetaFileBits Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryMetaFileBits - Parameters

hmf (HMF) - input
Memory-metafile handle.

lOffset (LONG) - input
Byte offset.

Offset into the metafile data from which the transfer must start. This is useful in instances where the metafile data is too long to fit into a single application buffer. It must be greater or equal to 0.

lLength (LONG) - input
Length in bytes of the metafile data to copy.

It must be greater or equal to 0.

pbData (PBYTE) - output
Metafile data.

Address in application storage into which the metafile data is copied.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiQueryMetaFileBits - Remarks

The total length of a metafile can be found from the data returned by GpiQueryMetaFileLength. This function allows an application to retrieve the data in units of a manageable size.

GpiQueryMetaFileBits - Errors

Possible returns from WinGetLastError

PMERR_INV_HMF (0x207E)
An invalid metafile handle was specified.

PMERR_INV_METAFILE_LENGTH (0x209E)
An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.

PMERR_INV_METAFILE_OFFSET (0x209F)
An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.

PMERR_METAFILE_IN_USE (0x20D9)
An attempt has been made to access a metafile that is in use by another thread.

PMERR_TOO_MANY_METAFILES_IN_USE (0x2106)
The maximum number of metafiles allowed for a given process was exceeded.

GpiQueryMetaFileBits - Example Code

This example uses the GpiQueryMetaFileBits function to retrieve the graphics-order data from the specified metafile. The GpiQueryMetaFileLength function returns the length of the metafile.

```
#define INCL_GPIMETAFILES       /* Metafile functions           */
#define INCL_DOSMEMMGR          /* DOS Memory Manager Functions */
#include <os2.h>

HPS hps;                /* presentation space handle           */
HMF hmf;                /* metafile handle                     */
LONG cBytes;            /* metafile length                     */
LONG off;               /* metafile byte offset                */
PBYTE pbBuffer;         /* metafile data buffer                */

hmf = GpiLoadMetaFile(hps, "sample.met");

cBytes = GpiQueryMetaFileLength(hmf);  /* gets length of metafile */

/* Allocate the buffer for the metafile data. */

DosAllocMem((VOID *)pbBuffer, (ULONG)cBytes,
            PAG_COMMIT | PAG_READ | PAG_WRITE);

GpiQueryMetaFileBits(
        hmf,          /* handle of metafile              */
        off,          /* offset of next byte to retrieve */
        cBytes,       /* length of data                  */
        pbBuffer);    /* buffer to receive metafile data */
```

GpiQueryMetaFileBits - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryMetaFileLength

GpiQueryMetaFileLength - Syntax

This function returns the total length of a memory metafile, in bytes.

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HMF     hmf;      /*  Memory-metafile handle. */
LONG    lLength;  /*  Total length of the metafile. */

lLength = GpiQueryMetaFileLength(hmf);
```

GpiQueryMetaFileLength Parameter - hmf

hmf (HMF) - input
Memory-metafile handle.

GpiQueryMetaFileLength Return Value - lLength

lLength (LONG) - returns
Total length of the metafile.

>=0

Total length of the metafile

GPI_ALTERROR
Error.

GpiQueryMetaFileLength - Parameters

hmf (HMF) - input
Memory-metafile handle.

lLength (LONG) - returns
Total length of the metafile.

>=0
Total length of the metafile

GPI_ALTERROR
Error.

GpiQueryMetaFileLength - Remarks

This function is normally used before GpiQueryMetaFileBits.

GpiQueryMetaFileLength - Errors

Possible returns from WinGetLastError

PMERR_INV_HMF (0x207E)
An invalid metafile handle was specified.

PMERR_TOO_MANY_METAFILES_IN_USE (0x2106)
The maximum number of metafiles allowed for a given process was exceeded.

GpiQueryMetaFileLength - Example Code

This example uses GpiQueryMetaFileLength to query the byte length of a memory metafile.

```
#define INCL_GPIMETAFILES       /* Meta File functions          */
#include <os2.h>

HMF    hmf;             /* memory-metafile handle               */
LONG  lLength;          /* length of metafile                   */

lLength = GpiQueryMetaFileLength(hmf);
```

GpiQueryMetaFileLength - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryMix

GpiQueryMix - Syntax

This function returns the current value of the (character) foreground color-mixing mode, as set by the GpiSetMix function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lMixMode;  /*  Mix mode. */

lMixMode = GpiQueryMix(hps);
```

GpiQueryMix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryMix Return Value - lMixMode

lMixMode (LONG) - returns
Mix mode.

FM_DEFAULT
(0L) Default
FM_OR
(1L) Logical-OR
FM_OVERPAINT
(2L) Overpaint
FM_XOR
(4L) Logical-XOR

FM_LEAVEALONE
(5L) Leave alone (invisible)
FM_AND
(6L) Logical-AND
FM_SUBTRACT
(7L) (Inverse source) AND destination
FM_MASKSRCNOT
(8L) Source AND (inverse destination)
FM_ZERO
(9L) All zeros
FM_NOTMERGESRC
(10L) Inverse (source OR destination)
FM_NOTXORSRC
(11L) Inverse (source XOR destination)
FM_INVERT
(12L) Inverse (destination)
FM_MERGESRCNOT
(13L) Source OR (inverse destination)
FM_NOTCOPYSRC
(14L) Inverse (source)
FM_MERGENOTSRC
(15L) (inveerse source) OR destination
FM_NOTMASKSRC
(16L) Inverse (source AND destination)
FM_ONE
(17L) All ones
FM_ERROR
(-1L) Error.

GpiQueryMix - Parameters

hps (HPS) - input
Presentation-space handle.

lMixMode (LONG) - returns
Mix mode.

FM_DEFAULT
(0L) Default
FM_OR
(1L) Logical-OR
FM_OVERPAINT
(2L) Overpaint
FM_XOR
(4L) Logical-XOR
FM_LEAVEALONE
(5L) Leave alone (invisible)
FM_AND
(6L) Logical-AND
FM_SUBTRACT
(7L) (Inverse source) AND destination
FM_MASKSRCNOT
(8L) Source AND (inverse destination)
FM_ZERO
(9L) All zeros
FM_NOTMERGESRC
(10L) Inverse (source OR destination)
FM_NOTXORSRC
(11L) Inverse (source XOR destination)
FM_INVERT
(12L) Inverse (destination)
FM_MERGESRCNOT
(13L) Source OR (inverse destination)
FM_NOTCOPYSRC
(14L) Inverse (source)

FM_MERGENOTSRC
(15L) (inveerse source) OR destination
FM_NOTMASKSRC
(16L) Inverse (source AND destination)
FM_ONE
(17L) All ones
FM_ERROR
(-1L) Error.

GpiQueryMix - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryMix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryMix - Example Code

This example uses GpiQueryMix to return the current foreground-mixing mode after setting the draw mode to DRAW.

```
#define INCL_GPIPRIMITIVES      /* Primitive functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

HPS    hps;             /* Presentation-space handle           */
LONG  lMixMode;         /* mix mode                            */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   lMixMode = GpiQueryMix(hps);
```

GpiQueryMix - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryModelTransformMatrix

GpiQueryModelTransformMatrix - Syntax

This function returns the current model transform; see GpiSetModelTransformMatrix.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
LONG         lCount;       /*  Number of elements. */
PMATRIXLF    pmatlfArray;  /*  Transform matrix. */
BOOL         rc;           /*  Success indicator. */

rc = GpiQueryModelTransformMatrix(hps, lCount,
       pmatlfArray);
```

GpiQueryModelTransformMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryModelTransformMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements to be returned in *pmatlfArray*; must be in the range 0 through 9. If 0 is specified, no matrix elements are returned.

GpiQueryModelTransformMatrix Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - output
Transform matrix.

A structure in which the elements of the model transform matrix are returned.

GpiQueryModelTransformMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryModelTransformMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of elements.

The number of elements to be returned in *pmatlfArray*; must be in the range 0 through 9. If 0 is specified, no matrix elements are returned.

pmatlfArray (PMATRIXLF) - output
Transform matrix.

A structure in which the elements of the model transform matrix are returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryModelTransformMatrix - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryModelTransformMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryModelTransformMatrix - Example Code

This example uses GpiQueryModelTransformMatrix to query the first element of the current model transform after setting the draw mode to DRAW.

```
#define INCL_GPITRANSFORMS      /* Transform functions          */
#define INCL_GPICONTROL         /* Control functions            */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
LONG  lCount;           /* number of elements                   */
MATRIXLF  pmatlfArray;  /* transform matrix                     */

lCount = 1; /* examine only first element of transform matrix */

if (GpiSetDrawingMode(hps, DM_DRAW) == TRUE)
   fSuccess = GpiQueryModelTransformMatrix(hps, lCount,
                                           &pmatlfArray);
```

GpiQueryModelTransformMatrix - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryNearestColor

GpiQueryNearestColor - Syntax

This function returns the nearest color available to the color specified on the currently associated device. Both colors are specified in RGB terms.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
ULONG    flOptions;  /*  Options. */
LONG     lRgbIn;     /*  Required color. */
LONG     lRgbOut;    /*  Nearest available color to the one specified. */

lRgbOut = GpiQueryNearestColor(hps, flOptions,
            lRgbIn);
```

GpiQueryNearestColor Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryNearestColor Parameter - flOptions

flOptions (ULONG) - input
Options.

Reserved, must be zero.

GpiQueryNearestColor Parameter - lRgbIn

lRgbIn (LONG) - input
Required color.

GpiQueryNearestColor Return Value - lRgbOut

lRgbOut (LONG) - returns
Nearest available color to the one specified.

>=0
: Nearest available color.

GPI_ALTERROR
: Error.

GpiQueryNearestColor - Parameters

hps (HPS) - input
: Presentation-space handle.

flOptions (ULONG) - input
: Options.

: Reserved, must be zero.

lRgbIn (LONG) - input
: Required color.

lRgbOut (LONG) - returns
: Nearest available color to the one specified.

: >=0
: Nearest available color

: GPI_ALTERROR
: Error.

GpiQueryNearestColor - Remarks

The nearest color returned is one that is available in the *physical* palette on the device. This might not actually be available with the currently loaded logical color table.

The color returned is a pure color, that is, one that can be used for drawing lines, text, and so on. It does not take into account the possibility of *dithered* colors being used for filled areas. With *dithering*, it is likely that the color used for filling areas is different from that used for lines, and text, when the same color index is selected.

For a monochrome device, if *lRgbIn* is the reset color, then *lRgbOut* is also the reset color; otherwise, it is black if the reset color is white, and the converse.

GpiQueryNearestColor - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COLOR_OPTIONS (0x2057)
: An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_RGBCOLOR (0x20C3)
: An invalid rgb color parameter was specified with GpiQueryNearestColor or GpiQueryColor.

GpiQueryNearestColor - Example Code

This example uses GpiQueryNearestColor to return the nearest color available to the one specified, on the currently associated device.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

LONG   lRgbOut;         /* nearest color                       */
HPS    hps;             /* Presentation-space handle           */
ULONG  ulOptions;       /* options                             */
LONG   lRgbIn;          /* color to match                      */

/* reserved; set to 0 */
ulOptions = 0L;

/* color to find index for (R= 36, G= 24, B= 37) */
lRgbIn = (36*65536) + (24*256) + 37;

lRgbOut = GpiQueryNearestColor(hps, ulOptions, lRgbIn);
```

GpiQueryNearestColor - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryNumberSetIds

GpiQueryNumberSetIds - Syntax

This function returns the number of local identifiers (lcids) currently in use, referring to fonts or bit maps.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lCount;  /*  Number of lcids. */

lCount = GpiQueryNumberSetIds(hps);
```

GpiQueryNumberSetIds Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryNumberSetIds Return Value - lCount

lCount (LONG) - returns
Number of lcids.

>=0
Number of lcids in use
GPI_ALTERROR
Error.

GpiQueryNumberSetIds - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - returns
Number of lcids.

>=0
Number of lcids in use
GPI_ALTERROR
Error.

GpiQueryNumberSetIds - Remarks

LCID_DEFAULT is included if the default font has been changed (see GpiCreateLogFont).

The information returned by this call can be used to perform a subsequent GpiQuerySetIds request.

GpiQueryNumberSetIds - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryNumberSetIds - Example Code

This example uses GpiQueryNumberSetIds to return the number of local identifiers in use (font and bit map).

```
#define INCL_GPILCIDS           /* Font functions               */
#include <os2.h>

LONG   lCount;          /* number of lcid's                    */
HPS    hps;             /* Presentation-space handle           */

lCount = GpiQueryNumberSetIds(hps);
```

GpiQueryNumberSetIds - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryPageViewport

GpiQueryPageViewport - Syntax

This function returns the page viewport; see GpiSetPageViewport.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;           /*  Presentation-space handle. */
PRECTL    prclViewport;  /*  Page viewport. */
BOOL      rc;            /*  Success indicator. */

rc = GpiQueryPageViewport(hps, prclViewport);
```

GpiQueryPageViewport Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPageViewport Parameter - prclViewport

prclViewport (PRECTL) - output
Page viewport.

The size and position of the page viewport in device units.

GpiQueryPageViewport Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryPageViewport - Parameters

hps (HPS) - input
Presentation-space handle.

prclViewport (PRECTL) - output
Page viewport.

The size and position of the page viewport in device units.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryPageViewport - Remarks

This function must not be issued when there is no device context associated with the presentation space.

GpiQueryPageViewport - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryPageViewport - Example Code

This example uses GpiQueryPageViewport to query the page viewport, after associating a device context to the presentation space; if successful, it assigns the x coordinate of the viewport to a variable.

```
#define INCL_GPITRANSFORMS      /* Transform functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
RECTL prclViewport;     /* page viewport                        */
LONG   lLwrLftxCoord;   /* lower left x coordinate of field     */
HDC hdc;                /* device context handle                */

/* associate device context */
if (GpiAssociate(hps, hdc) == TRUE)
   {
   fSuccess = GpiQueryPageViewport(hps, &prclViewport);

   /* if successful, assign lower left x coordinate of viewport */
   if (fSuccess == TRUE)
      lLwrLftxCoord = prclViewport.xLeft;
   }
```

GpiQueryPageViewport - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code

GpiQueryPalette

GpiQueryPalette - Syntax

This function returns the handle of the palette currently selected into a presentation space.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
HPAL    hpal;  /*  Palette handle. */

hpal = GpiQueryPalette(hps);
```

GpiQueryPalette Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPalette Return Value - hpal

hpal (HPAL) - returns
Palette handle.

NULLHANDLE
Null handle (no palette is selected)

PAL_ERROR
Error occurred

Otherwise
Handle of the palette currently selected into this presentation space.

GpiQueryPalette - Parameters

hps (HPS) - input

Presentation-space handle.

hpal (HPAL) - returns
Palette handle.

NULLHANDLE
Null handle (no palette is selected)
PAL_ERROR
Error occurred
Otherwise
Handle of the palette currently selected into this presentation space.

GpiQueryPalette - Remarks

It is possible for a palette to be selected into more than one presentation space at any one time. See GpiSelectPalette.

GpiQueryPalette - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryPalette - Example Code

This example uses GpiQueryPalette to return the handle of the palette currently selected into a presentation space and then calls GpiDeletePalette to delete the palette.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

HPAL   hpal;            /* palette handle                       */
HPS    hps;             /* Presentation-space handle            */
BOOL   fSuccess;        /* success indicator                    */

/* get handle of currently associated palette */
hpal = GpiQueryPalette(hps);

/* delete palette */
fSuccess = GpiDeletePalette(hpal);
```

GpiQueryPalette - Topics

Select an item:
Syntax
Parameters

Returns
Errors
Remarks
Example Code
Glossary

GpiQueryPaletteInfo

GpiQueryPaletteInfo - Syntax

This function passes back the information for a palette.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPAL      hpal;       /*  Palette handle. */
HPS       hps;        /*  Presentation-space handle. */
ULONG     flOptions;  /*  Specifies options. */
ULONG     ulStart;    /*  The starting index for which data is to be returned. */
ULONG     ulCount;    /*  Count of elements. */
PULONG    aulArray;   /*  An array in which the palette information is returned. */
LONG      lRetCount;  /*  Number of elements. */

lRetCount = GpiQueryPaletteInfo(hpal, hps,
              flOptions, ulStart, ulCount,
              aulArray);
```

GpiQueryPaletteInfo Parameter - hpal

hpal (HPAL) - input
Palette handle.

GpiQueryPaletteInfo Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPaletteInfo Parameter - flOptions

flOptions (ULONG) - input
Specifies options.

LCOLOPT_INDEX
If this is set, the index is to be returned for each RGB value in the *aulArray* parameter.

Other flags are reserved and must be 0.

GpiQueryPaletteInfo Parameter - ulStart

ulStart (ULONG) - input
The starting index for which data is to be returned.

GpiQueryPaletteInfo Parameter - ulCount

ulCount (ULONG) - input
Count of elements.

Number of elements available in *aulArray* .

If 0 is specified, the number of elements required to return the palette information in *aulArray* is returned.

GpiQueryPaletteInfo Parameter - aulArray

aulArray (PULONG) - output
An array in which the palette information is returned.

If LCOLOPT_INDEX is not specified, this is an array of RGB values (each value is as defined for GpiCreatePalette), starting with the specified index, and ending either when there are no further entries in the palette, or when *aulArray* has been exhausted. If the palette is not loaded with a contiguous set of indices, QLCT_NOTLOADED is returned as the RGB value for any index values, outside the default range, that have not been explicitly loaded.

If LCOLOPT_INDEX is specified, this is an array of alternating color indices and RGB values, in the order: index1, RGB value1, index2, RGB value2, ... An even number of elements is always returned. If the palette is not loaded with a contiguous set of indices, any index values that are not present are skipped.

GpiQueryPaletteInfo Return Value - lRetCount

lRetCount (LONG) - returns
Number of elements.

Zero is returned if no palette is selected.

PAL_ERROR
: Error occurred

Otherwise
: The number of elements of palette information passed back in the *aulArray* parameter, unless *ulCount* parameter is 0, in which case this is the total number of elements that are needed to hold the palette information.

GpiQueryPaletteInfo - Parameters

hpal (HPAL) - input
Palette handle.

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Specifies options.

LCOLOPT_INDEX
: If this is set, the index is to be returned for each RGB value in the *aulArray* parameter.

Other flags are reserved and must be 0.

ulStart (ULONG) - input
The starting index for which data is to be returned.

ulCount (ULONG) - input
Count of elements.

Number of elements available in *aulArray* .

If 0 is specified, the number of elements required to return the palette information in *aulArray* is returned.

aulArray (PULONG) - output
An array in which the palette information is returned.

If LCOLOPT_INDEX is not specified, this is an array of RGB values (each value is as defined for GpiCreatePalette), starting with the specified index, and ending either when there are no further entries in the palette, or when *aulArray* has been exhausted. If the palette is not loaded with a contiguous set of indices, QLCT_NOTLOADED is returned as the RGB value for any index values, outside the default range, that have not been explicitly loaded.

If LCOLOPT_INDEX is specified, this is an array of alternating color indices and RGB values, in the order: index1, RGB value1, index2, RGB value2, ... An even number of elements is always returned. If the palette is not loaded with a contiguous set of indices, any index values that are not present are skipped.

lRetCount (LONG) - returns
Number of elements.

Zero is returned if no palette is selected.

PAL_ERROR
: Error occurred

Otherwise
: The number of elements of palette information passed back in the *aulArray* parameter, unless *ulCount* parameter is 0, in which case this is the total number of elements that are needed to hold the palette information.

GpiQueryPaletteInfo - Remarks

The information passed back is in the same format as that required to create a palette (see GpiCreatePalette).

If a non-NULL palette handle is passed in the *hpal* parameter, the information is returned for that palette, and the *hps* parameter is ignored. Otherwise, *hps* identifies a presentation space for which the default colors are returned as a palette.

Note: In this case the default colors are returned, even if a logical color table is currently loaded into the presentation space.

GpiQueryPaletteInfo - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HPAL (0x2111)
: An invalid color palette handle was specified.

PMERR_INV_COLOR_OPTIONS (0x2057)
: An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_COLOR_START_INDEX (0x2058)
: An invalid starting index parameter was specified with a logical color table or color query function.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_PALETTE_BUSY (0x2112)
: An attempt has been made to reset the owner of a palette when it was busy.

GpiQueryPaletteInfo - Example Code

This example uses GpiQueryPaletteInfo to query the palette information and, if any values are returned, assigns the palette's first color value to a variable.

```
#define INCL_GPILOGCOLORTABLE   /* Color Table functions        */
#include <os2.h>

LONG  lRetCount;        /* number of elements                   */
HPAL  hpal;             /* palette handle                       */
ULONG flOptions;        /* options                              */
ULONG  ulStart;         /* starting index                       */
ULONG  ulCount;         /* count of elements in array           */
ULONG  *aulArray;       /* palette information array            */
ULONG  ulFirstColor;    /* first color in palette               */

/* specify no options */
flOptions = 0L;

/* start at index 0 */
ulStart = 0L;

/* tell function to determine element count */
ulCount = 0L;

lRetCount = GpiQueryPaletteInfo(hpal, NULLHANDLE, flOptions,
                                ulStart, ulCount,
                                aulArray);

/* if palette info returned, assign value of first color */
```

```
ulFirstColor = aulArray[0];
```

GpiQueryPaletteInfo - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryPattern

GpiQueryPattern - Syntax

This function returns the current value of the shading-pattern symbol, as set by the GpiSetPattern function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;            /*  Presentation-space handle. */
LONG    lPatternSymbol; /*  Pattern symbol. */

lPatternSymbol = GpiQueryPattern(hps);
```

GpiQueryPattern Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPattern Return Value - lPatternSymbol

lPatternSymbol (LONG) - returns
Pattern symbol.

PATSYM_DEFAULT
Default
PATSYM_DENSE1 to PATSYM_DENSE8
(1L to 8L) Solid shading with decreasing density
PATSYM_VERT
(9L) Vertical pattern
PATSYM_HORIZ
(10L) Horizontal pattern
PATSYM_DIAG1 to PATSYM_DIAG1
(11L to 14L) Diagonal pattern 1 to 4, bottom left to top right
PATSYM_NOSHADE
(15L) No shading
PATSYM_SOLID
(16L) Solid shading
PATSYM_HALFTONE
(17L) Alternate pels set on
PATSYM_BLANK
(18L) Blank
PATSYM_ERROR
(-1L) Error.

GpiQueryPattern - Parameters

hps (HPS) - input
Presentation-space handle.

lPatternSymbol (LONG) - returns
Pattern symbol.

PATSYM_DEFAULT
Default
PATSYM_DENSE1 to PATSYM_DENSE8
(1L to 8L) Solid shading with decreasing density
PATSYM_VERT
(9L) Vertical pattern
PATSYM_HORIZ
(10L) Horizontal pattern
PATSYM_DIAG1 to PATSYM_DIAG1
(11L to 14L) Diagonal pattern 1 to 4, bottom left to top right
PATSYM_NOSHADE
(15L) No shading
PATSYM_SOLID
(16L) Solid shading
PATSYM_HALFTONE
(17L) Alternate pels set on
PATSYM_BLANK
(18L) Blank
PATSYM_ERROR
(-1L) Error.

GpiQueryPattern - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryPattern - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryPattern - Example Code

In this example we query the current value of the shading-pattern symbol, as set by the GpiSetPattern call.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

LONG lResult;           /* pattern symbol if > 0     */
HPS hps;                /* Presentation space handle. */
if(PATSYM_SOLID == GpiQueryPattern(hps))
{
/* . */
/* . */
}
```

GpiQueryPattern - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryPatternRefPoint

GpiQueryPatternRefPoint - Syntax

This function returns the current pattern reference point, as set by the GpiSetPatternRefPoint function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;           /*  Presentation-space handle. */
PPOINTL    pptlRefPoint;  /*  Pattern reference point. */
BOOL       rc;            /*  Success indicator. */

rc = GpiQueryPatternRefPoint(hps, pptlRefPoint);
```

GpiQueryPatternRefPoint Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPatternRefPoint Parameter - pptlRefPoint

pptlRefPoint (PPOINTL) - output
Pattern reference point.

If the pattern reference point is currently set to the default, (0,0) is returned.

GpiQueryPatternRefPoint Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryPatternRefPoint - Parameters

hps (HPS) - input
Presentation-space handle.

pptlRefPoint (PPOINTL) - output
Pattern reference point.

If the pattern reference point is currently set to the default, (0,0) is returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryPatternRefPoint - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryPatternRefPoint - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryPatternRefPoint - Example Code

In this example we query the pattern reference point, which is set by the GpiSetPatternRefPoint.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
BOOL flResult;
HPS hps;                /* Presentation space handle. */
POINTL ptlRefPoint;     /* pattern reference point    */
LONG xcoord, ycoord;
flResult = GpiQueryPatternRefPoint(hps,
                                   &ptlRefPoint );

xcoord = ptlRefPoint.x; ycoord = ptlRefPoint.y;
```

GpiQueryPatternRefPoint - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryPatternSet

GpiQueryPatternSet - Syntax

This function returns the current value of the pattern-set identifier, as set by the GpiSetPatternSet function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
LONG    lSet;  /*  Pattern-set local identifier. */

lSet = GpiQueryPatternSet(hps);
```

GpiQueryPatternSet Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPatternSet Return Value - lSet

lSet (LONG) - returns
Pattern-set local identifier.

LCID_DEFAULT
Default
>0
Pattern set
LCID_ERROR
Error.

GpiQueryPatternSet - Parameters

hps (HPS) - input
Presentation-space handle.

lSet (LONG) - returns
Pattern-set local identifier.

LCID_DEFAULT
Default
>0
Pattern set
LCID_ERROR
Error.

GpiQueryPatternSet - Remarks

This function is not valid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryPatternSet - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryPatternSet - Example Code

In this example we query the pattern set identifier, which is set by the GpiSetPatternSet.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

LONG lpatternset;
HPS hps;                /* Presentation space handle. */

lpatternset = GpiQueryPatternSet(hps);
```

GpiQueryPatternSet - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryPel

GpiQueryPel - Syntax

This function returns the color of a pel at a position specified in world coordinates.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlPoint;  /*  Position in world coordinates. */
LONG       lColor;     /*  Color index of the pel. */

lColor = GpiQueryPel(hps, pptlPoint);
```

GpiQueryPel Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPel Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Position in world coordinates.

It is an error if the specified point is outside any of the current clipping objects (that is, clip path, viewing limits, clip region, or visible region).

GpiQueryPel Return Value - lColor

lColor (LONG) - returns
Color index of the pel.

This is a color index or RGB value, according to the logical color table in force (see GpiCreateLogColorTable).

>=0
Color of the pel
CLR_NOINDEX
No valid index (color is not in logical color table)
GPI_ALTERROR
Error.

GpiQueryPel - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - input
Position in world coordinates.

It is an error if the specified point is outside any of the current clipping objects (that is, clip path, viewing limits, clip region, or visible region).

lColor (LONG) - returns
Color index of the pel.

This is a color index or RGB value, according to the logical color table in force (see GpiCreateLogColorTable).

>=0
Color of the pel
CLR_NOINDEX
No valid index (color is not in logical color table)
GPI_ALTERROR
Error.

GpiQueryPel - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_PEL_IS_CLIPPED (0x20EF)
An attempt was made to query a pel that had been clipped using GpiQueryPel.

PMERR_PEL_NOT_AVAILABLE (0x20F0)
An attempt was made to query a pel that did not exist in GpiQueryPel (for example, a memory device context with no selected bit map).

PMERR_NO_BITMAP_SELECTED (0x20E4)
An attempt has been made to operate on a memory device context that has no bit map selected.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryPel - Example Code

In this example we query the color of a pel at a position specified in world coordinates.

```
#define INCL_GPIBITMAPS
#include <OS2.H>

LONG lcolorindex;       /* color index of pel. */
HPS hps;                /* Presentation space handle. */
POINTL     ptlPoint;   /* position in world coordinates. */
LONG xcoord, ycoord;
GpiQueryPel(hps, &ptlPoint);
xcoord = ptlPoint.x; ycoord = ptlPoint.y;
```

GpiQueryPel - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryPickAperturePosition

GpiQueryPickAperturePosition - Syntax

This function returns the position of the center of the pick aperture.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlPoint;  /*  Pick-aperture position. */
BOOL       rc;         /*  Success indicator. */

rc = GpiQueryPickAperturePosition(hps, pptlPoint);
```

GpiQueryPickAperturePosition Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPickAperturePosition Parameter - pptlPoint

pptlPoint (PPOINTL) - output
Pick-aperture position.

Position of the center of the pick aperture, in presentation-page coordinates.

GpiQueryPickAperturePosition Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryPickAperturePosition - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - output
Pick-aperture position.

Position of the center of the pick aperture, in presentation-page coordinates.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryPickAperturePosition - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryPickAperturePosition - Related Functions

Related Functions

- GpiQueryPickAperturePosition
- GpiQueryPickApertureSize
- GpiSetPickAperturePosition

GpiQueryPickAperturePosition - Example Code

In this example we query the position of the center of the pick aperture.

```
#define INCL_GPICORELATION
#include <OS2.H>

BOOL flResult;
HPS hps;               /* Presentation space handle. */
POINTL ptlRefPoint;   /* Pick-aperture position.    */
LONG xcoord, ycoord;
flResult = GpiPickAperturePosition(hps, &ptlRefPoint);
xcoord = ptlRefPoint.x; ycoord = ptlRefPoint.y;
```

GpiQueryPickAperturePosition - Topics

Select an item:
Syntax
Parameters

Returns
Errors
Example Code
Related Functions
Glossary

GpiQueryPickApertureSize

GpiQueryPickApertureSize - Syntax

This function returns the value of the pick-aperture size, as set by the GpiSetPickApertureSize function.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
PSIZEL    psizlSize;  /*  Pick-aperture size. */
BOOL      rc;         /*  Success indicator. */

rc = GpiQueryPickApertureSize(hps, psizlSize);
```

GpiQueryPickApertureSize Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPickApertureSize Parameter - psizlSize

psizlSize (PSIZEL) - output
Pick-aperture size.

Size of the pick aperture, in presentation-page coordinates.

GpiQueryPickApertureSize Return Value - rc

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryPickApertureSize - Parameters

hps (HPS) - input
: Presentation-space handle.

psizlSize (PSIZEL) - output
: Pick-aperture size.

Size of the pick aperture, in presentation-page coordinates.

rc (BOOL) - returns
: Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiQueryPickApertureSize - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryPickApertureSize - Example Code

In this example we query the pick-aperture size, as set by the GpiSetPickApertureSize call.

```
#define INCL_GPICORRELATION
#include <OS2.H>

BOOL flResult;
HPS hps;                /* Presentation space handle. */
SIZEL sizel;            /* Pick-aperture position.    */
LONG xcoord, ycoord;
flResult = GpiQueryPickApertureSize(hps, &sizel);
xcoord = sizel.cx; ycoord = sizel.cy;
```

GpiQueryPickApertureSize - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryPS

GpiQueryPS - Syntax

This function returns page parameters for the presentation space.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
PSIZEL    psizlSize;  /*  Presentation-page size. */
ULONG     lOptions;   /*  Presentation-space options. */

lOptions = GpiQueryPS(hps, psizlSize);
```

GpiQueryPS Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryPS Parameter - psizlSize

psizlSize (PSIZEL) - output
Presentation-page size.

GpiQueryPS Return Value - lOptions

lOptions (ULONG) - returns
Presentation-space options.

For details, see the GpiCreatePS function.

The individual fields of the presentation-space options can be extracted by ANDing the returned value with the appropriate constant.

The *PS_ASSOCIATE* field of *flOptions* (see GpiCreatePS) should not be used on this function. The value of this field is not necessarily the same value that is specified when the presentation space is created.

PS_UNITS
Presentation-space size units
PS_FORMAT
Presentation-space coordinate format
PS_TYPE
Presentation-space type
PS_MODE
Presentation-space mode.

GpiQueryPS - Parameters

hps (HPS) - input
Presentation-space handle.

psizlSize (PSIZEL) - output
Presentation-page size.

lOptions (ULONG) - returns
Presentation-space options.

For details, see the GpiCreatePS function.

The individual fields of the presentation-space options can be extracted by ANDing the returned value with the appropriate constant.

The *PS_ASSOCIATE* field of *flOptions* (see GpiCreatePS) should not be used on this function. The value of this field is not necessarily the same value that is specified when the presentation space is created.

PS_UNITS
Presentation-space size units
PS_FORMAT
Presentation-space coordinate format
PS_TYPE
Presentation-space type
PS_MODE
Presentation-space mode.

GpiQueryPS - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiQueryPS - Example Code

In this example we query the presentation space that corresponds to handle hps.

```
#define INCL_GPICONTROL
#include <OS2.H>
HPS hps;
SIZEL sizel;

GpiQueryPS(hps, &sizel);
```

GpiQueryPS - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryRealColors

GpiQueryRealColors - Syntax

This function returns the RGB values of the distinct colors available on the currently associated device.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
ULONG    flOptions;  /*  Options. */
LONG     lStart;     /*  Ordinal number of the first color required. */
LONG     lCount;     /*  Maximum number of elements. */
PLONG    alColors;   /*  Array in which the information is returned. */
LONG     lRetCount;  /*  Number of elements returned. */

lRetCount = GpiQueryRealColors(hps, flOptions,
              lStart, lCount, alColors);
```

GpiQueryRealColors Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryRealColors Parameter - flOptions

flOptions (ULONG) - input
Options.

LCOLOPT_INDEX
If this is specified, the index is to be returned for each RGB value.

If this flag is set when RGB mode is in force (LCOLF_RGB is set on GpiCreateLogColorTable), the RGB value is returned as the index.

Any color not available with the current logical color table is given a special index value of CLR_NOINDEX.

If it is not specified (flag is not set) index values are not returned.

Other
Other bits are reserved, and must be 0.

GpiQueryRealColors Parameter - lStart

lStart (LONG) - input
Ordinal number of the first color required.

To start the sequence, this parameter is set to 0.

Note: This parameter is not the color index, and the order in which the colors are returned is not defined.

GpiQueryRealColors Parameter - lCount

lCount (LONG) - input
Maximum number of elements.

Number of elements available in *alColors*. It must be greater or equal to 0.

GpiQueryRealColors Parameter - alColors

alColors (PLONG) - output
Array in which the information is returned.

Contents depend on the setting of the LCOLOPT_INDEX flag:

0
An array of color values (each value is as defined for GpiCreateLogColorTable).

1
An array of alternating color indexes and values, in the order index1, value1, index2, value2,... indexn, valuen. An even number of elements is always returned in this case.

GpiQueryRealColors Return Value - lRetCount

lRetCount (LONG) - returns
Number of elements returned.

>=0
Number of elements returned

GPI_ALTERROR
Error.

GpiQueryRealColors - Parameters

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Options.

LCOLOPT_INDEX
If this is specified, the index is to be returned for each RGB value.

If this flag is set when RGB mode is in force (LCOLF_RGB is set on GpiCreateLogColorTable), the RGB value is returned as the index.

Any color not available with the current logical color table is given a special index value of CLR_NOINDEX.

If it is not specified (flag is not set) index values are not returned.

Other
Other bits are reserved, and must be 0.

lStart (LONG) - input
Ordinal number of the first color required.

To start the sequence, this parameter is set to 0.

Note: This parameter is not the color index, and the order in which the colors are returned is not defined.

lCount (LONG) - input
Maximum number of elements.

Number of elements available in *alColors*. It must be greater or equal to 0.

alColors (PLONG) - output
Array in which the information is returned.

Contents depend on the setting of the LCOLOPT_INDEX flag:

0
: An array of color values (each value is as defined for GpiCreateLogColorTable).

1
: An array of alternating color indexes and values, in the order index1, value1, index2, value2,... indexn, valuen. An even number of elements is always returned in this case.

lRetCount (LONG) - returns
Number of elements returned.

>=0
: Number of elements returned

GPI_ALTERROR
: Error.

GpiQueryRealColors - Remarks

Subject to space in the *alColors* parameter, all colors that are physically available on the device are returned.

Use of the palette manager by other applications can effect the the physical colors available on the device. The available colors can change as a result of palette management, when this occurs a WM_REALIZEPALETTE message is sent to all applications.

GpiQueryRealColors - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_COLOR_OPTIONS (0x2057)
: An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_COLOR_START_INDEX (0x2058)
: An invalid starting index parameter was specified with a logical color table or color query function.

GpiQueryRealColors - Example Code

In this example we obtain the RGB values of the distinct colors available on the currently associated device.

```
#define INCL_GPILOGCOLORTABLE
#include <OS2.H>

LONG lResult;     /* number of elements returned */
HPS hps;          /* Presentation space handle. */
ULONG flOptions;  /* options                    */
LONG lStart;      /* ordinal number of first color */
LONG lCount;      /* maximum number of elements    */
LONG alColors[5];   /* array containing return information */
```

```
flOptions = LCOLOPT_INDEX; /* return index for each RGB value. */
lStart = 0L;               /* start sequence at 0.             */
lCount = 5L;               /* maximum of 5 elements.           */

lResult =  GpiQueryRealColors(hps,
                              flOptions,
                              lStart,
                              lCount,
                              alColors);
```

GpiQueryRealColors - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryRegionBox

GpiQueryRegionBox - Syntax

This function returns the dimensions of the smallest rectangle able to bound the region.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;          /*  Presentation-space handle. */
HRGN      hrgn;         /*  Region handle. */
PRECTL    prclBound;    /*  Bounding rectangle. */
LONG      lComplexity;  /*  Complexity of region and error indicators. */

lComplexity = GpiQueryRegionBox(hps, hrgn,
                prclBound);
```

GpiQueryRegionBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

GpiQueryRegionBox Parameter - hrgn

hrgn (HRGN) - input
Region handle.

GpiQueryRegionBox Parameter - prclBound

prclBound (PRECTL) - output
Bounding rectangle.

GpiQueryRegionBox Return Value - lComplexity

lComplexity (LONG) - returns
Complexity of region and error indicators.

RGN_NULL
Null region
RGN_RECT
Rectangular region
RGN_COMPLEX
Complex region
RGN_ERROR
Error.

GpiQueryRegionBox - Parameters

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) - input
Region handle.

prclBound (PRECTL) - output
Bounding rectangle.

lComplexity (LONG) - returns
Complexity of region and error indicators.

RGN_NULL
Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiQueryRegionBox - Remarks

If the region is null, the rectangle returned has the left boundary equal to the right, and the top boundary equal to the bottom.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiQueryRegionBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
: An invalid region handle was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
: An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY (0x2034)
: An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiQueryRegionBox - Example Code

In this example we determine the dimensions of the smallest rectangle able to bound the region.

```
#define INCL_GPIPREGIONS
#include <OS2.H>

LONG lResult;     /* number of elements returned */
HPS hps;          /* Presentation space handle. */
HRGN hrgn;        /* region handle */
RECTL rclBound; /* bounding rectangle */

lResult = GpiQueryRegionBox(hps,
                            (VOID *)hrgn,
                            (PRECTL)&rclBound);
```

GpiQueryRegionBox - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryRegionRects

GpiQueryRegionRects - Syntax

This function determines what rectangles that, when ORed together, define the specified region and then returns either the number of rectangles or the rectangles themselves.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS         hps;            /*  Presentation-space handle. */
HRGN        hrgn;           /*  Region handle. */
PRECTL      prclBound;      /*  Bounding rectangle. */
PRGNRECT    prgnrcControl;  /*  Processing-control structure. */
PRECTL      prclRect;       /*  Pointer to an array of rectangle structures. */
BOOL        rc;             /*  Success indicator. */

rc = GpiQueryRegionRects(hps, hrgn, prclBound,
       prgnrcControl, prclRect);
```

GpiQueryRegionRects Parameter - hps

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

GpiQueryRegionRects Parameter - hrgn

hrgn (HRGN) - input
Region handle.

GpiQueryRegionRects Parameter - prclBound

prclBound (PRECTL) - input
Bounding rectangle.

NULL
Return all the rectangles in the region.

Other
Return only rectangles that intersect with the bounding rectangle. Each rectangle returned is the intersection of the bounding rectangle with a rectangle in the region.

GpiQueryRegionRects Parameter - prgnrcControl

prgnrcControl (PRGNRECT) - in/out
Processing-control structure.

GpiQueryRegionRects Parameter - prclRect

prclRect (PRECTL) - output
Pointer to an array of rectangle structures or NULL.

NULL — Indicates only the number of rectanges within the region is to be returned.

The number of rectanges is returned in the *crcReturned* field of the RGNRECT structure identified by the *prgnrcControl* parameter. The *ircStart* field will always be 1. If a bounding rectangle is specified, the *crc* and *crcReturned* fields will be set as follows:

If the given rectangle is:	crc	crcReturned
Completely inside.	RRGN_INSIDE	1
Completely outside.	RRGN_OUTSIDE	0
Partially inside.	RRGN_PARTIAL	Number of rectangles

Other — A pointer to an array of rectangle structures in which the rectangles are returned.

The maximum number of rectangles that can be returned is specified by the *crc* parameter of the RGNRECT structure identified by the *prgnrcControl* parameter.

GpiQueryRegionRects Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryRegionRects - Parameters

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) - input
Region handle.

prclBound (PRECTL) - input
Bounding rectangle.

NULL
Return all the rectangles in the region.
Other
Return only rectangles that intersect with the bounding rectangle. Each rectangle returned is the intersection of the bounding rectangle with a rectangle in the region.

prgnrcControl (PRGNRECT) - in/out
Processing-control structure.

prclRect (PRECTL) - output
Pointer to an array of rectangle structures or NULL

NULL
Return the number of rectangles in the region.

The number of rectanges is returned in the *crcReturned* field of the RGNRECT structure identified by the *prgnrcControl* parameter. The *ircStart* field will always be 1. If a bounding rectangle is specified, the *crc* and *crcReturned* fields will be set as follows:

If the given rectangle is:	crc	crcReturned
Completely inside.	RRGN_INSIDE	1
Completely outside.	RRGN_OUTSIDE	0
Partially inside.	RRGN_PARTIAL	Number of rectangles

Other
Return only rectangles that intersect with the bounding rectangle. Each rectangle returned is the intersection of the bounding rectangle with a rectangle in the region.

Pointer to an array of rectangle structures in which the rectangles are returned.

The maximum number of rectangles that can be returned is specified by the *crc* parameter of the RGNRECT structure identified by the *prgnrcControl* parameter.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryRegionRects - Remarks

Points on the right-hand and top boundaries are not included in the region. Points on the left-hand and bottom boundaries, that are not also on the right-hand or top boundaries (that is, the top-left and bottom-right corner points), are included.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiQueryRegionRects - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_INV_REGION_CONTROL (0x20BE)
An invalid control parameter was specified with GpiQueryRegionRects.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
An invalid rectangle parameter was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiQueryRegionRects - Example Code

In this example we determine the rectangles that can be OR'ed together to determine the specified region.

```
#define INCL_GPIREGIONS
#include <OS2.H>
#define MAXRECTS 12

BOOL flResult;      /* success indicator.         */
HPS hps;            /* presentation space handle. */
HRGN hrgn;          /* region handle.             */
```

```
RECTL rclBound;     /* bounding rectangle         */
RGNRECT rgnrcControl;  /* processing control      */
RECTL arclRect[MAXRECTS];  /* array of rectangle structures */
                           /* in which the rectangles are   */
                           /* returned.                     */

rgnrcControl.ircStart = 1;    /* start numbering rectangles */
                              /* from 1.                    */
rgnrcControl.crc = MAXRECTS;  /* maximum number of rectangles */
                              /* that can be returned.        */
rgnrcControl.usDirection = RECTDIR_LFRT_TOPBOT;
                               /* order rectangles left-to-right */
                               /* and top-to-bottom.             */
flResult =  GpiQueryRegionRects(hps,
                                 hrgn,
                                 &rclBound,      /* output */
                                 &rgnrcControl,
                               /* rgnrcControl.crcReturned is the */
                               /* number of rectangles returned.  */
                                 &arclRect[0]);
```

GpiQueryRegionRects - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryRGBColor

GpiQueryRGBColor - Syntax

This function returns the actual RGB color that results from a particular index on the currently-associated device.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;          /*  Presentation-space handle. */
ULONG    flOptions;    /*  Options. */
LONG     lColorIndex;  /*  Color index. */
LONG     lRgbColor;    /*  RGB color providing closest match to the specified color index. */

lRgbColor = GpiQueryRGBColor(hps, flOptions,
              lColorIndex);
```

GpiQueryRGBColor Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryRGBColor Parameter - flOptions

flOptions (ULONG) - input
Options.

LCOLOPT_REALIZED
If this is specified, the information is required for when the logical color table is realized.

If it is not specified (flag is not set) the information is required for when the logical color table (if any) is not realized.

Other bits are reserved, and must be 0.

GpiQueryRGBColor Parameter - lColorIndex

lColorIndex (LONG) - input
Color index.

This can be any normally valid color index value (see GpiSetColor) except CLR_DEFAULT. When using the system color table, it can also be any SYSCLR_* value.

GpiQueryRGBColor Return Value - lRgbColor

lRgbColor (LONG) - returns
RGB color providing closest match to the specified color index.

>=0
RGB color providing closest match
GPI_ALTERROR
Error.

GpiQueryRGBColor - Parameters

hps (HPS) - input
Presentation-space handle.

flOptions (ULONG) - input
Options.

LCOLOPT_REALIZED
If this is specified, the information is required for when the logical color table is realized.

If it is not specified (flag is not set) the information is required for when the logical color table (if any) is not realized.

Other bits are reserved, and must be 0.

lColorIndex (LONG) - input
Color index.

This can be any normally valid color index value (see GpiSetColor) except CLR_DEFAULT. When using the system color table, it can also be any SYSCLR_* value.

lRgbColor (LONG) - returns
RGB color providing closest match to the specified color index.

>=0
RGB color providing closest match
GPI_ALTERROR
Error.

GpiQueryRGBColor - Remarks

If an RGB logical color table has been loaded, this function returns the nearest RGB color. This function is then equivalent to GpiQueryNearestColor.

GpiQueryRGBColor - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COLOR_OPTIONS (0x2057)
An invalid options parameter was specified with a logical color table or color query function.

PMERR_INV_COLOR_INDEX (0x2056)
An invalid color index parameter was specified with GpiQueryRGBColor.

GpiQueryRGBColor - Example Code

This example uses the GpiQueryRGBColor call to determine if the color white is available.

```
#define INCL_GPILOGCOLORTABLE
#include <OS2.H>

LONG lResult;     /* closest match to the specified index */
HPS hps;          /* Presentation space handle. */
```

```
ULONG flOptions;  /* options                    */
LONG lColorIndex; /* color index                */
lColorIndex = CLR_WHITE;
flOptions = LCOLOPT_REALIZED;
                  /* information is required for when the */
                  /* logical color table is realized.     */

lResult = GpiQueryRGBColor(hps,
                           flOptions,
                           lColorIndex );
```

GpiQueryRGBColor - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQuerySegmentAttrs

GpiQuerySegmentAttrs - Syntax

This function returns the current value of the specified attribute as set by the GpiSetSegmentAttrs function.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;         /*  Presentation-space handle. */
LONG    lSegid;      /*  Segment identifier; must be greater than 0. */
LONG    lAttribute;  /*  Attribute to be queried. */
LONG    lValue;      /*  Current attribute value. */

lValue = GpiQuerySegmentAttrs(hps, lSegid,
           lAttribute);
```

GpiQuerySegmentAttrs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQuerySegmentAttrs Parameter - lSegid

lSegid (LONG) - input
Segment identifier; must be greater than 0.

The name of the segment for which attribute information is to be returned.

GpiQuerySegmentAttrs Parameter - lAttribute

lAttribute (LONG) - input
Attribute to be queried.

For details of the following attributes, see the GpiSetInitialSegmentAttrs function.

Identifies the attribute of the segment to be returned by this function:

ATTR_DETECTABLE
Detectability
ATTR_VISIBLE
Visibility
ATTR_CHAINED
Chained
ATTR_DYNAMIC
Dynamic
ATTR_FASTCHAIN
Fast chaining
ATTR_PROP_DETECTABLE
Propagate detectability
ATTR_PROP_VISIBLE
Propagate visibility.

GpiQuerySegmentAttrs Return Value - lValue

lValue (LONG) - returns
Current attribute value.

ATTR_ON
On/yes
ATTR_OFF
Off/no
ATTR_ERROR
Error.

GpiQuerySegmentAttrs - Parameters

hps (HPS) - input
Presentation-space handle.

lSegid (LONG) - input
Segment identifier; must be greater than 0.

The name of the segment for which attribute information is to be returned.

lAttribute (LONG) - input
Attribute to be queried.

For details of the following attributes, see the GpiSetInitialSegmentAttrs function.

Identifies the attribute of the segment to be returned by this function:

ATTR_DETECTABLE
Detectability
ATTR_VISIBLE
Visibility
ATTR_CHAINED
Chained
ATTR_DYNAMIC
Dynamic
ATTR_FASTCHAIN
Fast chaining
ATTR_PROP_DETECTABLE
Propagate detectability
ATTR_PROP_VISIBLE
Propagate visibility.

lValue (LONG) - returns
Current attribute value.

ATTR_ON
On/yes
ATTR_OFF
Off/no
ATTR_ERROR
Error.

GpiQuerySegmentAttrs - Remarks

The segment can be any retained segment, including the currently open one if this is retained.

GpiQuerySegmentAttrs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_SEG_ATTR (0x20C5)
An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQuerySegmentAttrs - Example Code

This function is used to query the current value of the specified attribute.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

LONG lSegid;     /* Segment identifier must       */
LONG lValue;     /* be greater than 0.            */
LONG lattribute; /* attribute to be queried       */
HPS hps;         /*   Presentation-space          */
                 /*   handle.                     */

lattribute = ATTR_VISIBLE;

lValue = GpiQuerySegmentAttrs(hps,
                              lSegid,
                              lattribute);
```

GpiQuerySegmentAttrs - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQuerySegmentNames

GpiQuerySegmentNames - Syntax

This function returns the identifiers of all segments that exist with identifiers in a specified range.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;          /*  Presentation-space handle. */
```

```
LONG     lFirstSegid;  /*  First segment in the range (must be greater than 0). */
LONG     lLastSegid;   /*  Last segment in the range (must be greater than 0). */
LONG     lMax;         /*  Maximum number. */
PLONG    alSegids;     /*  Array in which the required identifiers are returned. */
LONG     lRetCount;    /*  Number of identifiers returned. */

lRetCount = GpiQuerySegmentNames(hps, lFirstSegid,
              lLastSegid, lMax, alSegids);
```

GpiQuerySegmentNames Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQuerySegmentNames Parameter - lFirstSegid

lFirstSegid (LONG) - input
First segment in the range (must be greater than 0).

GpiQuerySegmentNames Parameter - lLastSegid

lLastSegid (LONG) - input
Last segment in the range (must be greater than 0).

GpiQuerySegmentNames Parameter - lMax

lMax (LONG) - input
Maximum number.

This is the maximum number of segment identifiers to be returned in *alSegids* .

GpiQuerySegmentNames Parameter - alSegids

alSegids (PLONG) - output
Array in which the required identifiers are returned.

GpiQuerySegmentNames Return Value - lRetCount

lRetCount (LONG) - returns
Number of identifiers returned.

>=0
Number of identifiers returned
GPI_ALTERROR
Error.

GpiQuerySegmentNames - Parameters

hps (HPS) - input
Presentation-space handle.

lFirstSegid (LONG) - input
First segment in the range (must be greater than 0).

lLastSegid (LONG) - input
Last segment in the range (must be greater than 0).

lMax (LONG) - input
Maximum number.

This is the maximum number of segment identifiers to be returned in *alSegids* .

alSegids (PLONG) - output
Array in which the required identifiers are returned.

lRetCount (LONG) - returns
Number of identifiers returned.

>=0
Number of identifiers returned
GPI_ALTERROR
Error.

GpiQuerySegmentNames - Remarks

Nonretained segment identifiers are not returned. If *lFirstSegid* is the same as, or greater than *lLastSegid* , the search terminates after querying only the segment with *lFirstSegid* .

GpiQuerySegmentNames - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQuerySegmentNames - Example Code

This function returns the identifiers of all segments that exist within a specified range.

```
#define INCL_GPISEGMENTS
#include <OS2.H>
#define Maxsegs 5

LONG lRetCount;
HPS hps;  /*   Presentation-space */
          /*   handle.            */
LONG lFirstSegid;   /*  First segment in the   */
                    /*  range (must be greater */
                    /*  than 0).               */

LONG lLastSegid;    /*  Last segment in the    */
                    /*  range (must be greater */
                    /*  than 0).               */

LONG lMax;          /*  This is the maximum        */
                    /*  number of segment          */
                    /*  identifiers to be returned */
                    /*  in alSegids.               */
LONG alSegids[Maxsegs];      /*  Array in which the         */
                             /*  required identifiers are   */
                             /*  returned.                  */
lFirstSegid = 1;
lLastSegid = Maxsegs;
lMax = Maxsegs;

lRetCount = GpiQuerySegmentNames(hps,
                                 lFirstSegid,
                                 lLastSegid,
                                 lMax,
                                 alSegids);
```

GpiQuerySegmentNames - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQuerySegmentPriority

GpiQuerySegmentPriority - Syntax

This function returns the identifier of the named segment that is chained immediately before or after a specified reference segment.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;        /*  Presentation-space handle. */
LONG    lRefSegid;  /*  Reference-segment identifier. */
LONG    lOrder;     /*  Segment higher or lower. */
LONG    lSegid;     /*  Segment identifier. */

lSegid = GpiQuerySegmentPriority(hps, lRefSegid,
           lOrder);
```

GpiQuerySegmentPriority Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQuerySegmentPriority Parameter - lRefSegid

lRefSegid (LONG) - input
Reference-segment identifier.

GpiQuerySegmentPriority Parameter - lOrder

lOrder (LONG) - input
Segment higher or lower.

Shows whether a segment identifier of a higher or lower priority than identified in the *lRefSegid* parameter is to be returned. Possible values are:

LOWER_PRI
Return the next segment with a lower priority than *lRefSegid*. If *lRefSegid*=0, query the identifier of the segment

with the lowest priority.

HIGHER_PRI
Return the next segment with a higher priority than *lRefSegid*. If *lRefSegid*=0, query the identifier of the segment with the highest priority.

GpiQuerySegmentPriority Return Value - lSegid

lSegid (LONG) - returns
Segment identifier.

The identifier of the segment that is immediately before or after that specified in the *lRefSegid* parameter:

>0
Segment identifier.

0
The segment specified in the *lRefSegid* parameter is either the lowest-priority segment (when *lOrder* = LOWER_PRI) or the highest-priority segment (when *lOrder* = HIGHER_PRI).

GPI_ALTERROR
Error.

GpiQuerySegmentPriority - Parameters

hps (HPS) - input
Presentation-space handle.

lRefSegid (LONG) - input
Reference-segment identifier.

lOrder (LONG) - input
Segment higher or lower.

Shows whether a segment identifier of a higher or lower priority than identified in the *lRefSegid* parameter is to be returned. Possible values are:

LOWER_PRI
Return the next segment with a lower priority than *lRefSegid*. If *lRefSegid*=0, query the identifier of the segment with the lowest priority.

HIGHER_PRI
Return the next segment with a higher priority than *lRefSegid*. If *lRefSegid*=0, query the identifier of the segment with the highest priority.

lSegid (LONG) - returns
Segment identifier.

The identifier of the segment that is immediately before or after that specified in the *lRefSegid* parameter:

>0
Segment identifier.

0
The segment specified in the *lRefSegid* parameter is either the lowest-priority segment (when *lOrder* = LOWER_PRI) or the highest-priority segment (when *lOrder* = HIGHER_PRI).

GPI_ALTERROR

Error.

GpiQuerySegmentPriority - Remarks

The segment that is chained before the specified segment, is considered to have a lower priority than the specified segment; similarly, the segment that is chained after the specified segment, is considered to have a higher priority than the specified segment.

Unnamed segments (with an identifier of zero) are ignored.

GpiQuerySegmentPriority - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_ORDERING_PARM (0x20AB)
An invalid order parameter was specified with GpiSetSegmentPriority.

PMERR_SEG_NOT_CHAINED (0x20FF)
An attempt was made to issue GpiDrawFrom, GpiCorrelateFrom or GpiQuerySegmentPriority for a segment that was not chained.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQuerySegmentPriority - Example Code

This function returns the identifier of the named segment that is chained immediately before or after a specified reference segment.

```
#define INCL_GPISEGMENTS
#include <OS2.H>


HPS hps;         /* Presentation-space        */
                 /* handle.                   */
LONG lSegid;     /* Segment identifier        */
LONG lRefSegid;  /* Reference-segment         */
                 /* identifier.               */
LONG lOrder;     /* Shows whether a           */
                 /* segment identifier of a   */
                 /* higher or lower priority  */
                 /* than identified in the    */
                 /*   lRefSegid parameter is    */
                 /*   to be returned.           */

lOrder =  HIGHER_PRI;  /* Return the next        */
                       /* segment with a higher  */
```

```
                     /* priority than          */
                     /* lRefSegid. If          */
                     /* lRefSegid=0, query     */
                     /* the identifier of the  */
                     /* segment with the       */
                     /* highest priority.      */
lRefSegid = 0;  /* find the segment with the highest */
                /* priority.                      */

lSegid = GpiQuerySegmentPriority(hps,
                                 lRefSegid,
                                 lOrder);
```

GpiQuerySegmentPriority - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQuerySegmentTransformMatrix

GpiQuerySegmentTransformMatrix - Syntax

This function returns the elements of the transform of the identified segment (see GpiSetSegmentTransformMatrix).

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
LONG         lSegid;       /*  Segment identifier. */
LONG         lCount;       /*  Number of elements. */
PMATRIXLF    pmatlfArray;  /*  Transform matrix. */
BOOL         rc;           /*  Success indicator. */

rc = GpiQuerySegmentTransformMatrix(hps, lSegid,
       lCount, pmatlfArray);
```

GpiQuerySegmentTransformMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQuerySegmentTransformMatrix Parameter - lSegid

lSegid (LONG) - input
Segment identifier.

GpiQuerySegmentTransformMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements that are to be set in the *pmatlfArray* parameter. *lCount* must be in the range 0 through 9.

GpiQuerySegmentTransformMatrix Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - output
Transform matrix.

GpiQuerySegmentTransformMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQuerySegmentTransformMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lSegid (LONG) - input

Segment identifier.

lCount (LONG) - input
Number of elements.

The number of elements that are to be set in the *pmatlfArray* parameter. *lCount* must be in the range 0 through 9.

pmatlfArray (PMATRIXLF) - output
Transform matrix.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQuerySegmentTransformMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

GpiQuerySegmentTransformMatrix - Example Code

This function returns the elements of the transform of the identified segment (see GpiSetSegmentTransformMatrix).

```
#define INCL_GPITRANSFORMS   /* Or use INCL_GPI or INCL_PM */
#include<OS2.H>
#define COUNT 9

HPS hps;         /*   Presentation-space */
                 /*   handle.            */
LONG lSegid;     /*   Segment identifier. */
LONG lCount;     /* The number of elements     */
                 /* that are to be set in the  */
                 /* pmatlfArray parameter.     */
                 /* lCount must be in the      */
                 /* range 0 through 9.         */
MATRIXLF pmatlfArray[COUNT];   /* array of Transform matrix  */
                               /* structures. This is an     */
                               /* output parameter.          */
BOOL fSuccess;                 /* returns true if successful. */
```

```
fSuccess = GpiQuerySegmentTransformMatrix(hps,
                                          lSegid,
                                          lCount,
                                          pmatlfArray);
```

GpiQuerySegmentTransformMatrix - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQuerySetIds

GpiQuerySetIds - Syntax

This function returns information about all the fonts that have been created by GpiCreateLogFont, and tagged bit maps (see GpiSetBitmapId).

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;      /*  Presentation-space handle. */
LONG     lCount;   /*  The number of objects to be queried. */
PLONG    alTypes;  /*  Object types. */
PSTR8    aNames;   /*  Font names. */
PLONG    allcids;  /*  Local identifiers. */
BOOL     rc;       /*  Success indicator. */

rc = GpiQuerySetIds(hps, lCount, alTypes,
       aNames, allcids);
```

GpiQuerySetIds Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQuerySetIds Parameter - lCount

lCount (LONG) - input
The number of objects to be queried.

The number of local identifiers (lcids) currently in use, and therefore the maximum number of objects for which information can be returned, can be found with GpiQueryNumberSetIds.

GpiQuerySetIds Parameter - alTypes

alTypes (PLONG) - output
Object types.

Elements indicate whether the corresponding *allcids* element refers to a logical font or a tagged bit map.

LCIDT_FONT
Font object
LCIDT_BITMAP
Bit map.

GpiQuerySetIds Parameter - aNames

aNames (PSTR8) - output
Font names.

An array of *lCount* consecutive 8-byte fields, in which the 8-character names associated with the logical fonts are returned. For bit maps, the whole field is set to zeros.

GpiQuerySetIds Parameter - allcids

allcids (PLONG) - output
Local identifiers.

An array in which the local identifier (lcid) values are returned.

LCID_DEFAULT is included if the default font has been changed (see GpiCreateLogFont).

GpiQuerySetIds Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQuerySetIds - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
The number of objects to be queried.

The number of local identifiers (lcids) currently in use, and therefore the maximum number of objects for which information can be returned, can be found with GpiQueryNumberSetIds.

alTypes (PLONG) - output
Object types.

Elements indicate whether the corresponding *allcids* element refers to a logical font or a tagged bit map.

LCIDT_FONT
Font object
LCIDT_BITMAP
Bit map.

aNames (PSTR8) - output
Font names.

An array of *lCount* consecutive 8-byte fields, in which the 8-character names associated with the logical fonts are returned. For bit maps, the whole field is set to zeros.

allcids (PLONG) - output
Local identifiers.

An array in which the local identifier (lcid) values are returned.

LCID_DEFAULT is included if the default font has been changed (see GpiCreateLogFont).

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQuerySetIds - Remarks

Each of the output parameters is an array with *lCount* elements. Information about the first *lCount* objects is returned; if there are fewer than *lCount*, the *alTypes* and *allcids* elements for the remainder are cleared to 0.

GpiQuerySetIds - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

GpiQuerySetIds - Example Code

This example uses the GpiQuerySetIds function to retrieve the local identifier for all logical fonts. It then uses the identifiers to delete the logical fonts.

```
#define  INCL_DOSMEMMGR
#define  INCL_GPILCIDS
#include <OS2.H>
#define  TOTALMEM 200

HPS    hps;      /* Presentation-space handle.           */
LONG   lCount;   /* The number of objects to be queried. */
PLONG  alTypes;  /* Object types.                        */
ULONG  rc;       /* Return code.                         */
PSTR8  aNames;   /* font names.                          */
PLONG  allcids;  /* local identifiers.                   */
PLONG  pBase;
USHORT i;

rc = DosAllocMem((PPVOID)pBase,
                 (ULONG)TOTALMEM*sizeof(LONG),
                          /* space is needed for an array of  */
                          /* lCount longs.                    */
                  PAG_READ |
                  PAG_WRITE |
                  PAG_COMMIT);

lCount =  GpiQueryNumberSetIds(hps);
                        /*  The number of local identifiers  */
                        /*  (lcids) currently in use, and    */
                        /*  therefore the maximum number     */
                        /*  of objects for which information */
                        /*  can be returned.                 */

rc = DosSubAllocMem((PVOID)pBase,
                   (PPVOID)aNames,
                    (ULONG)(lCount*(ULONG)sizeof(STR8)));
                        /* space is needed for an array of   */
                        /* lCount longs.                     */
rc = DosSubAllocMem((PVOID)pBase,
                    (PPVOID)allcids,
                    (ULONG)lCount*sizeof(LONG));
                        /* space is needed for an array of   */
                        /* lCount longs.                     */
rc = DosSubAllocMem((PVOID)pBase,
                    (PPVOID)alTypes,
                    (ULONG)lCount*sizeof(LONG));
                        /* space is needed for an array of   */
                        /* lCount longs.                     */

           GpiQuerySetIds(hps,
                          lCount,
                          alTypes,
                          aNames, /*  An array of lCount         */
                                  /*  consecutive 8-byte fields, */
                                  /*  in which the 8-character   */
                                  /*  names associated with      */
```

```
                                  /*  the logical fonts are      */
                                  /*  returned. For bit maps,    */
                                  /*  the whole field is set to  */
                                  /*  zeros.                     */
                          allcids);/*  An array in which the      */
                                  /*  local identifier (lcid)    */
                                  /*  values are returned.       */
                                  /*  LCID_DEFAULT is         */
                                  /*  included if the default    */
                                  /*  font has been changed      */
                                  /*  (see GpiCreateLogFont).    */

for (i = 1; i < lCount; i++)
{
 if (allcids[i] == LCIDT_FONT)
  GpiDeleteSetId(hps,allcids[i]);
}
```

GpiQuerySetIds - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryStopDraw

GpiQueryStopDraw - Syntax

This function indicates whether the "stop draw" condition currently exists.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lValue;  /*  Stop draw condition indicator. */

lValue = GpiQueryStopDraw(hps);
```

GpiQueryStopDraw Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryStopDraw Return Value - lValue

lValue (LONG) - returns
Stop draw condition indicator.

SDW_OFF
No "stop draw" condition currently exists
SDW_ON
The "stop draw" condition currently exists
SDW_ERROR
Error.

GpiQueryStopDraw - Parameters

hps (HPS) - input
Presentation-space handle.

lValue (LONG) - returns
Stop draw condition indicator.

SDW_OFF
No "stop draw" condition currently exists
SDW_ON
The "stop draw" condition currently exists
SDW_ERROR
Error.

GpiQueryStopDraw - Remarks

See GpiSetStopDraw for details.

GpiQueryStopDraw - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQueryStopDraw - Example Code

This function indicates whether the "stop draw" condition currently exists.

```
#define INCL_GPICONTROL
#include <OS2.H>
HPS hps;         /*   Presentation-space  handle. */
LONG lValue;

if(GpiQueryStopDraw(hps) == SDW_OFF)
{
    /* drawing may proceed; no stop draw */
    /* condition exists.                 */
}
```

GpiQueryStopDraw - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryTag

GpiQueryTag - Syntax

This function returns the current value of the tag identifier, as set by the GpiSetTag function.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;    /*  Presentation-space handle. */
PLONG    plTag;  /*  Tag identifier. */
BOOL     rc;     /*  Success indicator. */

rc = GpiQueryTag(hps, plTag);
```

GpiQueryTag Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryTag Parameter - plTag

plTag (PLONG) - output
Tag identifier.

GpiQueryTag Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryTag - Parameters

hps (HPS) - input
Presentation-space handle.

plTag (PLONG) - output
Tag identifier.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryTag - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryTag - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiQueryTag - Example Code

This function returns the current value of the tag identifier, as set by the GpiSetTag call.

```
#define INCL_GPICORRELATION
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

LONG lTag;      /*   Tag identifier.    */


GpiQueryTag(hps,
            &lTag);
```

GpiQueryTag - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryTextAlignment

GpiQueryTextAlignment - Syntax

This function returns the current value of the text alignment attribute, as set by the GpiSetTextAlignment function.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;      /*  Presentation-space handle. */
PLONG    plHoriz;  /*  Horizontal alignment. */
PLONG    plVert;   /*  Vertical alignment. */
BOOL     rc;       /*  Success indicator. */

rc = GpiQueryTextAlignment(hps, plHoriz, plVert);
```

GpiQueryTextAlignment Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryTextAlignment Parameter - plHoriz

plHoriz (PLONG) - output
Horizontal alignment.

The horizontal alignment determines character positioning in a text string. The value returned will be one of those described under the GpiSetTextAlignment function.

GpiQueryTextAlignment Parameter - plVert

plVert (PLONG) - output
Vertical alignment.

The vertical alignment determines character positioning in a text string. The value returned will be one of those described under the GpiSetTextAlignment function.

GpiQueryTextAlignment Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryTextAlignment - Parameters

hps (HPS) - input
Presentation-space handle.

plHoriz (PLONG) - output
Horizontal alignment.

The horizontal alignment determines character positioning in a text string. The value returned will be one of those described under the GpiSetTextAlignment function.

plVert (PLONG) - output
Vertical alignment.

The vertical alignment determines character positioning in a text string. The value returned will be one of those described under the GpiSetTextAlignment function.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryTextAlignment - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

Support for this function is device dependent.

GpiQueryTextAlignment - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryTextAlignment - Related Functions

Related Functions

- GpiQueryAttrs
- GpiSetTextAlignment

GpiQueryTextAlignment - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Related Functions
Glossary

GpiQueryTextBox

GpiQueryTextBox - Syntax

This function returns the relative coordinates of the four corners of a text box.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lCount1;     /*  Number of characters. */
PCH        pchString;   /*  The character string. */
LONG       lCount2;     /*  Number of points. */
PPOINTL    aptlPoints;  /*  List of points. */
BOOL       rc;          /*  Success indicator. */

rc = GpiQueryTextBox(hps, lCount1, pchString,
       lCount2, aptlPoints);
```

GpiQueryTextBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryTextBox Parameter - lCount1

lCount1 (LONG) - input
Number of characters.

It must be greater or equal to 0 and less or equal to 512

GpiQueryTextBox Parameter - pchString

pchString (PCH) - input
The character string.

GpiQueryTextBox Parameter - lCount2

lCount2 (LONG) - input
Number of points.

Contains the number of points to be returned in the *aptlPoints* array. Specify TXTBOX_COUNT to get the maximum information. This parameter must be greater or equal to 0.

GpiQueryTextBox Parameter - aptlPoints

aptlPoints (PPOINTL) - output
List of points.

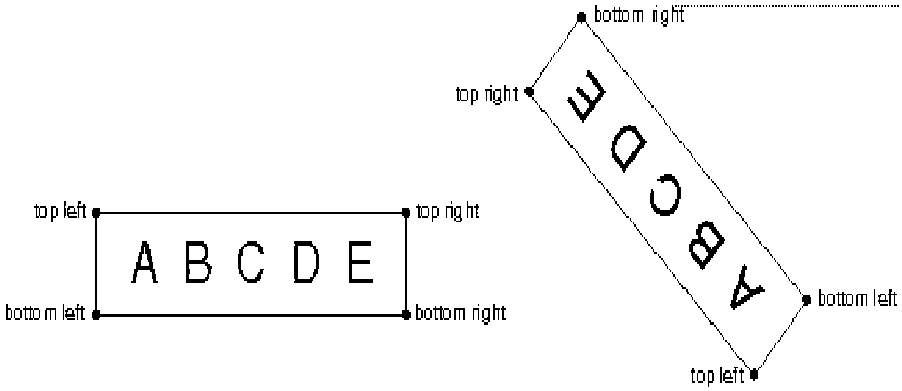
The list of points contains the relative coordinates of the text box in world coordinates. The array elements are numbered consecutively, starting with TXTBOX_TOPLEFT. The element number constants start with 0. A *lCount2* value of TXTBOX_COUNT will cause all of the defined array elements to be returned.

The terms 'top-left', 'bottom-right', and so on, are well defined when the character angle is such that the baseline is parallel to the x axis and running left to right, and there is no character shear. If the character string is rotated or sheared, the term top-left applies to the corner of the box that appears in the top-left position when no rotation or shear is applied.

This is an example:

```
Set character angle = -1,1
String = ABCDE
```

Coordinates returned are as shown:



The possible values for this parameter are:

TXTBOX_TOPLEFT
Top-left corner
TXTBOX_BOTTOMLEFT
Bottom-left corner
TXTBOX_TOPRIGHT
Top-right corner
TXTBOX_BOTTOMRIGHT
Bottom-right corner
TXTBOX_CONCAT
Concatenation point.

GpiQueryTextBox Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryTextBox - Parameters

hps (HPS) - input
Presentation-space handle.

lCount1 (LONG) - input
Number of characters.

It must be greater or equal to 0 and less or equal to 512

pchString (PCH) - input
The character string.

lCount2 (LONG) - input
Number of points.

Contains the number of points to be returned in the *aptlPoints* array. Specify TXTBOX_COUNT to get the maximum information. This parameter must be greater or equal to 0.

aptlPoints (PPOINTL) - output
List of points.

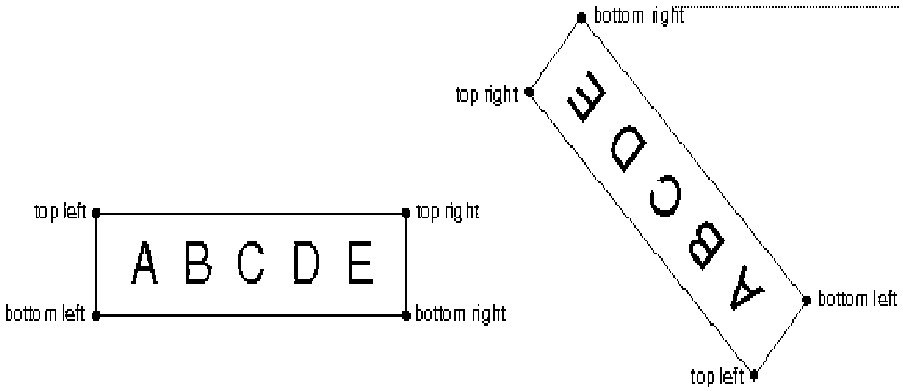
The list of points contains the relative coordinates of the text box in world coordinates. The array elements are numbered consecutively, starting with TXTBOX_TOPLEFT. The element number constants start with 0. A *lCount2* value of TXTBOX_COUNT will cause all of the defined array elements to be returned.

The terms 'top-left', 'bottom-right', and so on, are well defined when the character angle is such that the baseline is parallel to the x axis and running left to right, and there is no character shear. If the character string is rotated or sheared, the term top-left applies to the corner of the box that appears in the top-left position when no rotation or shear is applied.

This is an example:

```
Set character angle = -1,1
String = ABCDE
```

Coordinates returned are as shown:



The possible values for this parameter are:

TXTBOX_TOPLEFT
Top-left corner
TXTBOX_BOTTOMLEFT
Bottom-left corner
TXTBOX_TOPRIGHT
Top-right corner
TXTBOX_BOTTOMRIGHT
Bottom-right corner
TXTBOX_CONCAT
Concatenation point.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryTextBox - Remarks

The text box is defined as the parallelogram that encloses the specified character string when displayed on the device. Also returned are the relative coordinates of the concatenation point; that is, the value of current position after an equivalent GpiCharStringAt function. All coordinates are relative to the start point. (See GpiSetCharDirection function.) These coordinates can be used to box or underline the string, or to change the attributes in the middle of a longer string.

Note: The height of the string is based on the maximum height of the font (including space for descenders, accents, and so on), not the maximum height of the actual characters in the string. The dimensions of the box do not correspond directly to those of the character box (see GpiSetCharBox).

Character attributes are taken into account as if the string is to be drawn, but no output actually occurs. However, if the character mode (see GpiSetCharMode) is CM_MODE2 this function should only be used if the character angle (see GpiSetCharAngle), character direction (see GpiSetCharDirection) and character shear (see GpiSetCharShear) attributes are set to their default values.

This function is not valid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryTextBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
: An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW (0x2014)
: An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryTextBox - Example Code

This example uses the GpiQueryTextBox function to draw a line under the string. The GpiCharString function draws the string at the point (100,100). Since the points retrieved by GpiQueryTextBox are relative to the start of the string, the starting point needs to be added to the points that are used to draw the underline.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;
POINTL aptl[TXTBOX_COUNT];
POINTL ptl = { 100, 100 };

GpiQueryTextBox(hps,
                11L,
                "This string",
                TXTBOX_COUNT,  /* return maximum information */
                aptl);         /* array of coordinates points */
                               /* in world coordinates.       */
aptl[1].x += ptl.x;
aptl[1].y += ptl.y;
GpiMove(hps, &aptl[1]);
aptl[3].x += ptl.x;
aptl[3].y += ptl.y;
GpiLine(hps, &aptl[3]);
GpiMove(hps, &ptl);
GpiCharString(hps, 11L, "This string");
```

GpiQueryTextBox - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryViewingLimits

GpiQueryViewingLimits - Syntax

This function returns the current value of the viewing limits, as set by the GpiSetViewingLimits function.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;         /*  Presentation-space handle. */
PRECTL    prclLimits;  /*  Viewing limits. */
BOOL      rc;          /*  Success indicator. */

rc = GpiQueryViewingLimits(hps, prclLimits);
```

GpiQueryViewingLimits Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryViewingLimits Parameter - prclLimits

prclLimits (PRECTL) - output
Viewing limits.

GpiQueryViewingLimits Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryViewingLimits - Parameters

hps (HPS) - input
Presentation-space handle.

prclLimits (PRECTL) - output
Viewing limits.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryViewingLimits - Remarks

This function is invalid when the drawing mode (see GpiSetDrawingMode) is set to **retain**.

GpiQueryViewingLimits - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_IN_RETAIN_MODE (0x208C)
An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.

PMERR_INV_DC_TYPE (0x2060)
An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiQueryViewingLimits - Example Code

In this example the model space clipping region width is reduced to 100 if it is greater.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */
RECTL rclLimits; /* viewing limits.    */
BOOL fSuccess;


fSuccess = GpiQueryViewingLimits(hps,
                                 &rclLimits);

if ((rclLimits.xRight - rclLimits.xLeft) > 100)
{
 rclLimits.xRight = 100;
 rclLimits.xLeft = 200;
}

fSuccess = GpiSetViewingLimits(hps,
                               &rclLimits);
```

GpiQueryViewingLimits - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiQueryViewingTransformMatrix

GpiQueryViewingTransformMatrix - Syntax

This function returns the current viewing transform (see GpiSetViewingTransformMatrix)..

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
LONG         lCount;       /*  Number of elements. */
PMATRIXLF    pmatlfArray;  /*  Transform matrix. */
BOOL         rc;           /*  Success indicator. */
```

```
rc = GpiQueryViewingTransformMatrix(hps, lCount,
       pmatlfArray);
```

GpiQueryViewingTransformMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryViewingTransformMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements to be returned in *pmatlfArray* (must be in the range 0 through 9). If 0 is specified, no matrix elements are returned.

GpiQueryViewingTransformMatrix Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - output
Transform matrix.

A structure in which the elements of the viewing transform matrix are returned.

GpiQueryViewingTransformMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryViewingTransformMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of elements.

The number of elements to be returned in *pmatlfArray* (must be in the range 0 through 9). If 0 is specified, no matrix elements are returned.

pmatlfArray (PMATRIXLF) - output
Transform matrix.

A structure in which the elements of the viewing transform matrix are returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryViewingTransformMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

GpiQueryViewingTransformMatrix - Example Code

This example uses the GpiQueryViewingTransformMatrix function to see if the width and the height of drawing are already doubled. If this is not the case, the GpiSetViewingTransformMatrix is used to replace the existing viewing transformation. The new transformation will then double the width and height of drawing.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;          /* Presentation space handle. */
LONG lCount;      /* maximum number of elements */
MATRIXLF matlf = { MAKEFIXED(2,0),  /* scale x coordinates by a */
                                    /* factor of 2.             */
                   0, 0, 0,         /* no rotation.             */
                   MAKEFIXED(2,0),  /* scale y coordinates by a */
                                    /* factor of 2.             */
                   0, 0, 0, 1};     /* no rotation.             */
lCount = 9L;                        /* number of elements.      */
GpiQueryViewingTransformMatrix(hps,
                               lCount,
                               &matlf);

if (matlf.fxM12 == MAKEFIXED(2, 0))
```

```
{
GpiSetViewingTransformMatrix(hps,
                             lCount,
                             &matlf,
                             TRANSFORM_REPLACE);

}
```

GpiQueryViewingTransformMatrix - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiQueryWidthTable

GpiQueryWidthTable - Syntax

This function returns width table information for the logical font identified by the value of the character-set attribute.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;         /*  Presentation-space handle. */
LONG     lFirstChar;  /*  Codepoint of first character. */
LONG     lCount;      /*  Count of elements in alData. */
PLONG    alData;      /*  Array of width values. */
BOOL     rc;          /*  Success indicator. */

rc = GpiQueryWidthTable(hps, lFirstChar, lCount,
       alData);
```

GpiQueryWidthTable Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiQueryWidthTable Parameter - lFirstChar

lFirstChar (LONG) - input
Codepoint of first character.

The codepoint of the initial character, for which width-table information is required.

GpiQueryWidthTable Parameter - lCount

lCount (LONG) - input
Count of elements in *alData* .

The number that should be allowed for, so as to retrieve the full width table. Data for this font can be found by GpiQueryFontMetrics.

GpiQueryWidthTable Parameter - alData

alData (PLONG) - output
Array of width values.

An array of *lCount* elements, in which width-table information is returned. No more than *lCount* elements are returned.

GpiQueryWidthTable Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryWidthTable - Parameters

hps (HPS) - input
Presentation-space handle.

lFirstChar (LONG) - input
Codepoint of first character.

The codepoint of the initial character, for which width-table information is required.

lCount (LONG) - input
Count of elements in *alData* .

The number that should be allowed for, so as to retrieve the full width table. Data for this font can be found by GpiQueryFontMetrics.

alData (PLONG) - output
Array of width values.

An array of *lCount* elements, in which width-table information is returned. No more than *lCount* elements are returned.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiQueryWidthTable - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_FIRST_CHAR (0x2071)
An invalid firstchar parameter was specified with GpiQueryWidthTable.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_COORDINATE_OVERFLOW (0x2014)
An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.

GpiQueryWidthTable - Example Code

In this example the widths of the first 50 characters of the current font are obtained.

```
#define INCL_GPILCIDS
#include <OS2.H>
#define COUNT 50


HPS hps;              /*   Presentation-space */
                      /*   handle.            */
LONG alData[COUNT];   /* array of width values. */


GpiQueryWidthTable(hps,
                   0,
                   COUNT,
                   alData);
```

GpiQueryWidthTable - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Glossary

GpiRectInRegion

GpiRectInRegion - Syntax

This function checks whether any part of a rectangle lies within the specified region.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;       /*  Presentation-space handle. */
HRGN      hrgn;      /*  Region handle. */
PRECTL    prclRect;  /*  Test rectangle. */
LONG      lInside;   /*  Inside and error indicators. */

lInside = GpiRectInRegion(hps, hrgn, prclRect);
```

GpiRectInRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

GpiRectInRegion Parameter - hrgn

hrgn (HRGN) - input
Region handle.

GpiRectInRegion Parameter - prclRect

prclRect (PRECTL) - input
Test rectangle.

The rectangle is specified in device coordinates.

GpiRectInRegion Return Value - lInside

lInside (LONG) - returns
Inside and error indicators.

RRGN_OUTSIDE
Not in region
RRGN_PARTIAL
Some in region
RRGN_INSIDE
All in region
RRGN_ERROR
Error.

GpiRectInRegion - Parameters

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) - input
Region handle.

prclRect (PRECTL) - input
Test rectangle.

The rectangle is specified in device coordinates.

lInside (LONG) - returns
Inside and error indicators.

RRGN_OUTSIDE
Not in region
RRGN_PARTIAL
Some in region
RRGN_INSIDE
All in region
RRGN_ERROR
Error.

GpiRectInRegion - Remarks

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiRectInRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
An invalid rectangle parameter was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiRectInRegion - Related Functions

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiSetRegion

GpiRectInRegion - Example Code

In this example we check to see if a a rectangle is inside a region before we destroy the region.

```
#define INCL_GPIREGIONS
#include <OS2.H>
HPS hps;         /* presentation-space handle. */
HRGN hrgn;       /* region handle. */
PRECTL prclRect; /* test rectangle. */
LONG lInside;    /* result.         */
lInside = GpiRectInRegion(hps,
```

```
                        hrgn,
                        prclRect);

if (lInside == RRGN_OUTSIDE)
{
GpiDestroyRegion(hps, hrgn);
}
```

GpiRectInRegion - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiRectVisible

GpiRectVisible - Syntax

This function checks whether any part of a rectangle lies within the clipping region of the device associated with the specified presentation space.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;            /*  Presentation-space handle. */
PRECTL    prclRectangle;  /*  Test rectangle, in world coordinates. */
LONG      lVisibility;    /*  Visibility indicator. */

lVisibility = GpiRectVisible(hps, prclRectangle);
```

GpiRectVisible Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiRectVisible Parameter - prclRectangle

prclRectangle (PRECTL) - input
Test rectangle, in world coordinates.

Points on the borders of the rectangle are considered to be included within the rectangle.

GpiRectVisible Return Value - lVisibility

lVisibility (LONG) - returns
Visibility indicator.

RVIS_INVISIBLE
Not visible
RVIS_PARTIAL
Some of the rectangle is visible
RVIS_VISIBLE
All of the rectangle is visible
RVIS_ERROR
Error.

GpiRectVisible - Parameters

hps (HPS) - input
Presentation-space handle.

prclRectangle (PRECTL) - input
Test rectangle, in world coordinates.

Points on the borders of the rectangle are considered to be included within the rectangle.

lVisibility (LONG) - returns
Visibility indicator.

RVIS_INVISIBLE
Not visible
RVIS_PARTIAL
Some of the rectangle is visible
RVIS_VISIBLE
All of the rectangle is visible
RVIS_ERROR
Error.

GpiRectVisible - Remarks

For the purposes of this function, the clipping region is defined as the intersection between the application clipping region and any other clipping, including windowing.

GpiRectVisible - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
: An invalid rectangle parameter was specified.

GpiRectVisible - Related Functions

Related Functions

- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiPtVisible
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipRegion

GpiRectVisible - Example Code

In this example the GpiRectVisible call is used to determine if all of the rectangle is visible.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;                /* presentation-space handle. */
LONG lVisibility;       /* visibility indicator       */
PRECTL prclRectangle;   /* test rectangle in world    */
                        /* coordinates.               */


lVisibility = GpiRectVisible(hps,
                             prclRectangle);

if (lVisibility == RVIS_INVISIBLE)  /* rectangle is not */
{                               /* visible.         */
 /* code block */
}
```

GpiRectVisible - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiRemoveDynamics

GpiRemoveDynamics - Syntax

This function removes those parts of the displayed image that are drawn from the dynamic segments in a section of the segment chain. This includes any parts that are drawn by calls from these dynamic segments.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;          /*  Presentation-space handle. */
LONG    lFirstSegid;  /*  First segment in the section. */
LONG    lLastSegid;   /*  Last segment in the section. */
BOOL    rc;           /*  Success indicator. */

rc = GpiRemoveDynamics(hps, lFirstSegid, lLastSegid);
```

GpiRemoveDynamics Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiRemoveDynamics Parameter - lFirstSegid

lFirstSegid (LONG) - input
First segment in the section.

It must be greater than 0.

GpiRemoveDynamics Parameter - lLastSegid

lLastSegid (LONG) - input
Last segment in the section.

It must be greater than 0.

GpiRemoveDynamics Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiRemoveDynamics - Parameters

hps (HPS) - input
Presentation-space handle.

lFirstSegid (LONG) - input
First segment in the section.

It must be greater than 0.

lLastSegid (LONG) - input
Last segment in the section.

It must be greater than 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiRemoveDynamics - Remarks

This function usually indicates that a dynamic segment is about to be updated; and that, having completed the update, GpiDrawDynamics is called to redraw the dynamic segments.

If there is more than one dynamic segment, only those that are being updated need be removed. The section of the segment chain is identified by the first and last segments in the section. If *lFirstSegid* and *lLastSegid* have the same value, this call erases only the parts

drawn from the segment, and by calls from that segment.

Specifying the range of segment identifiers that are to be removed usually has a performance advantage, in that searching of the chain stops after *lLastSegid* has been processed. It can also be used to operate on less than the maximum number of dynamic segments, as in one of the following examples:

- Several dynamic segments are currently drawn, but only one is to be updated. Identifying this segment with both *lFirstSegid* and *lLastSegid* means that only this one is removed. It can then be updated, and replaced with GpiDrawDynamics.
- A new dynamic segment can be created, while the rest remain drawn. GpiRemoveDynamics is issued before the segment has been created (or while it is unchained, if it already exists), identifying it with both *lFirstSegid* and *lLastSegid*. It is then created with this identifier (or chained, if it already exists), and GpiDrawDynamics is issued, causing it to be drawn.

In these examples, the other dynamic segments remain drawn throughout.

In all cases, after GpiDrawChain, GpiDrawDynamics, GpiDrawFrom, or GpiDrawSegment, where the DCTL_DYNAMIC draw control is set (see GpiSetDrawControl), *all* dynamic segments must be drawn. The *lFirstSegid* and *lLastSegid* parameters of GpiRemoveDynamics, cannot be used to cause a subset of dynamic segments to be drawn after the following GpiDrawDynamics. If this is required, it can be done by unchaining the unwanted dynamic segments after first removing them.

Dynamic segments that are currently drawn must never be updated in the segment store, nor must any drawing in mix modes (other than exclusive-OR or leave-alone) be done to a presentation space while dynamic segments are drawn in it.

If a temporary re-association is to be done, this function must be issued to remove the dynamic segments from the display before the first dissociation.

It is necessary to ensure that attributes, model transform, current position, and viewing limits are reset to their default values, before processing the segments. This can either be done by ensuring that the first dynamic segment in the removed section does not have the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), or by issuing GpiResetPS before the GpiRemoveDynamics. The latter method also resets the clip path to cause no clipping, which may also be necessary.

If this function is followed by primitives or attributes, without first opening a segment, the processing is as described for GpiCloseSegment. In particular, note that during GpiRemoveDynamics, the system forces the foreground mix to FM_XOR and the background mix to BM_LEAVEALONE. It may be necessary to set one or both of these before starting to draw.

If *lFirstSegid* does not exist, or is not in the segment chain, no removal or drawing occurs. However, the segment identifier range is still established for a subsequent GpiDrawDynamics function.

If *lLastSegid* does not exist, or is not in the chain, or is chained before *lFirstSegid*, no error is raised, and processing continues to the end of the chain.

GpiRemoveDynamics - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_FOR_THIS_DC_TYPE (0x2074)
An attempt has been made to issue GpiRemoveDynamics or GpiDrawDynamics to a presentation space associated with a metafile device context.

GpiRemoveDynamics - Related Functions

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiErase
- GpiGetData
- GpiPutData
- GpiSetDrawControl
- GpiSetDrawingMode
- GpiSetStopDraw

GpiRemoveDynamics - Example Code

This example uses the GpiRemoveDynamics function to remove the image drawn by the dynamic segment whose segment identifier is 4. It then edits the segment and redraws it, using the GpiDrawDynamics function.

```
#define INCL_GPISEGMENTS
#define INCL_GPICONTROL
#include <OS2.H>


 POINTL ptl = {30, 40};
 HPS hps; /* presentation space handle */

 /* Remove the image for dynamic segment #4. */

 GpiRemoveDynamics(hps, 4L, 4L);

 /* Edit the segment. */

 GpiSetDrawingMode(hps,DM_RETAIN);
 GpiOpenSegment(hps, 4L);
 GpiSetElementPointer(hps, 1L);
 GpiMove(hps, &ptl);
 GpiCloseSegment(hps);

 GpiDrawDynamics(hps);    /* redraws the edited segment */
```

GpiRemoveDynamics - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiResetBoundaryData

GpiResetBoundaryData - Syntax

This function resets the boundary data to null.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;  /*  Presentation-space handle. */
BOOL    rc;   /*  Success indicator. */

rc = GpiResetBoundaryData(hps);
```

GpiResetBoundaryData Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiResetBoundaryData Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiResetBoundaryData - Parameters

hps (HPS) - input
Presentation-space handle.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiResetBoundaryData - Remarks

This function is only necessary for **draw** mode (see GpiSetDrawingMode) boundary determination. Boundary data is automatically reset before any retained drawing call.

After drawing, boundary data can be found by issuing GpiQueryBoundaryData.

Note: Boundary data is not reset at the start of a segment.

GpiResetBoundaryData - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiResetBoundaryData - Related Functions

Related Functions

- GpiQueryBoundaryData
- GpiSetDrawControl

GpiResetBoundaryData - Example Code

This function is used to reset the boundary data to null. It is only necessary for draw mode boundary determination.

```
#define INCL_GPICORRELATION
#include <OS2.H>

HPS hps;         /* presentation space handle */


GpiResetBoundaryData(hps);
```

GpiResetBoundaryData - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiResetPS

GpiResetPS - Syntax

This function resets the presentation space.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
ULONG    flOptions;  /*  Reset option. */
BOOL     rc;         /*  Success indicator. */

rc = GpiResetPS(hps, flOptions);
```

GpiResetPS Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiResetPS Parameter - flOptions

flOptions (ULONG) - input
Reset option.

GRES_ATTRS
This has the following effects:

- All current attributes and arc parameters are reset to their default values
- The current tag is reset to its default value
- The current model transform is reset to unity
- The current position is set to (0,0)
- Any open path or area is aborted
- Any open element bracket is closed

- Any open segment is closed
- The current clip path is set so as to cause no clipping
- The current viewing limits are reset to their default values.

GRES_SEGMENTS
: This has all the effects of GRES_ATTRS plus:

- Any retained segments are deleted
- Initial segment attributes are reset to their default values
- The default viewing transform and the graphics field are reset to their default values
- The viewing transform is set to unity
- Drawing mode, draw controls, edit mode and attribute mode are reset to default values
- Boundary data is reset
- The currently selected clip region, if any, is deselected, and destroyed
- The default values of primitive attributes, arc parameters, viewing limits and primitive tag are reset to their initial values.

GRES_ALL
: This has all the effects of GRES_ATTRS and GRES_SEGMENTS plus:

- Any logical fonts and local identifiers for bit maps are deleted (the default character set is restored if it has been changed).
- Any loaded logical color table is reset to default.
- Any palette selected into the presentation space (see GpiSelectPalette) is deselected.
- The pick aperture size and position are reset to default.

GpiResetPS Return Value - rc

rc (BOOL) - returns
: Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiResetPS - Parameters

hps (HPS) - input
: Presentation-space handle.

flOptions (ULONG) - input
: Reset option.

GRES_ATTRS
: This has the following effects:

- All current attributes and arc parameters are reset to their default values
- The current tag is reset to its default value
- The current model transform is reset to unity
- The current position is set to (0,0)
- Any open path or area is aborted
- Any open element bracket is closed
- Any open segment is closed
- The current clip path is set so as to cause no clipping
- The current viewing limits are reset to their default values.

GRES_SEGMENTS

This has all the effects of GRES_ATTRS plus:

- Any retained segments are deleted
- Initial segment attributes are reset to their default values
- The default viewing transform and the graphics field are reset to their default values
- The viewing transform is set to unity
- Drawing mode, draw controls, edit mode and attribute mode are reset to default values
- Boundary data is reset
- The currently selected clip region, if any, is deselected, and destroyed
- The default values of primitive attributes, arc parameters, viewing limits and primitive tag are reset to their initial values.

GRES_ALL
This has all the effects of GRES_ATTRS and GRES_SEGMENTS plus:

- Any logical fonts and local identifiers for bit maps are deleted (the default character set is restored if it has been changed).
- Any loaded logical color table is reset to default.
- Any palette selected into the presentation space (see GpiSelectPalette) is deselected.
- The pick aperture size and position are reset to default.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiResetPS - Remarks

Three levels of reset are provided. These are, in increasing order of power:

- As if a new (root) segment is being processed
- As if the presentation space is being created without deleting resources
- As if the presentation space is being created with resources deleted.

More details are provided under the description of *flOptions* above.

None of these options cause any drawing or erasing to take place on the device (GpiErase can be used to do the latter), nor is any association between the specified presentation space and a device context affected. The page viewport is also unaffected.

After restoring a presentation space that has a palette selected into it, WinRealizePalette must be issued before any drawing calls or calls to query colors.

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiResetPS - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_RESET_OPTIONS (0x20C2)

An invalid options parameter was specified with GpiResetPS.

GpiResetPS - Related Functions

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiRestorePS
- GpiSavePS
- GpiSetPS

GpiResetPS - Example Code

This function is used to reset the presentation space.

```
#define INCL_GPICONTROL
#include <OS2.H>

HPS hps;         /* presentation space handle */
ULONG flOptions; /* reset options */

flOptions = GRES_ALL;  /* reset all options. */

GpiResetPS(hps, flOptions);
```

GpiResetPS - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiRestorePS

GpiRestorePS - Syntax

This function restores the state of the presentation space to the one that exists when the corresponding GpiSavePS is issued.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lPSid;  /*  Identifier of the saved presentation space that is to be restored. */
BOOL    rc;     /*  Success indicator. */

rc = GpiRestorePS(hps, lPSid);
```

GpiRestorePS Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiRestorePS Parameter - lPSid

lPSid (LONG) - input
Identifier of the saved presentation space that is to be restored.

If an error is returned, the stack is unchanged, as is the current presentation space.

>0	*lPSid* must be the identifier of a saved presentation space on the stack. It is an error if it does not exist.
0	Is an error. (This might have resulted from an invalid use of GpiSavePS).
<0	The absolute value of *lPSid* indicates how many saved presentation spaces on the stack are required. Thus -1 means that the most recently saved one is to be restored. It is an error if the absolute value is larger than the number of entries on the stack.

GpiRestorePS Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE	Successful completion
FALSE	Error occurred.

GpiRestorePS - Parameters

hps (HPS) - input
Presentation-space handle.

lPSid (LONG) - input
Identifier of the saved presentation space that is to be restored.

If an error is returned, the stack is unchanged, as is the current presentation space.

>0	*lPSid* must be the identifier of a saved presentation space on the stack. It is an error if it does not exist.
0	Is an error. (This might have resulted from an invalid use of GpiSavePS).
<0	The absolute value of *lPSid* indicates how many saved presentation spaces on the stack are required. Thus -1 means that the most recently saved one is to be restored. It is an error if the absolute value is larger than the number of entries on the stack.

rc (BOOL) - returns
Success indicator.

TRUE	Successful completion
FALSE	Error occurred.

GpiRestorePS - Remarks

The most recently saved presentation space need not be the one that is restored. In this case, any that are skipped over on the stack are discarded.

Any clip regions selected into discarded presentation spaces are automatically destroyed.

This function is valid in an open element bracket and in an open segment bracket if the drawing mode (see GpiSetDrawingMode) is set to **draw** and within an open element bracket. If it occurs within an open area or path bracket, the corresponding GpiSavePS must have taken place earlier in the same bracket.

It is recommended that GpiSavePS and GpiRestorePS are used in pairs and are NOT split across page boundaries in a multiple-page print job.

GpiRestorePS - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_DRAW_MODE (0x20DE)
An attempt was made to issue GpiSavePS or GpiRestorePS while the drawing mode was not set to DM_DRAW.

PMERR_INV_ID (0x2081)
An invalid *lPSid* parameter was specified with GpiRestorePS.

GpiRestorePS - Related Functions

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiSavePS
- GpiSetPS
- GpiPop

GpiRestorePS - Example Code

This example restores the state of the presentation space to the one that exists when the corresponding GpiSavePS is issued.

```
#define INCL_GPICONTROL
#include <OS2.H>
HPS hps;      /* presentation space handle */
LONG lPSid;   /* the identifier of a saved presentation */
              /* space on the stack.                    */
GpiRestorePS(hps, lPSid);
```

GpiRestorePS - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiRotate

GpiRotate - Syntax

This function applies a rotation to a transform matrix.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
PMATRIXLF    pmatlfArray;  /*  Transform matrix. */
LONG         lOptions;     /*  Transform options. */
FIXED        fxAngle;      /*  Rotation angle. */
PPOINTL      pptlCenter;   /*  Center of rotation. */
BOOL         rc;           /*  Success indicator. */

rc = GpiRotate(hps, pmatlfArray, lOptions,
       fxAngle, pptlCenter);
```

GpiRotate Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiRotate Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - in/out
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

GpiRotate Parameter - lOptions

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the specified rotation should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE
The previous transform is discarded and replaced by the transform describing the specified rotation.

TRANSFORM_ADD
The previous transform is combined with a transform representing the specified rotation in the order (1) previous transform, (2) rotational transform. This option is most useful for incremental updates to transforms.

GpiRotate Parameter - fxAngle

fxAngle (FIXED) - input
Rotation angle.

The angle describing the rotation, measured counterclockwise from the x-axis in degrees.

GpiRotate Parameter - pptlCenter

pptlCenter (PPOINTL) - input
Center of rotation.

The point about which the rotation occurs.

GpiRotate Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiRotate - Parameters

hps (HPS) - input
Presentation-space handle.

pmatlfArray (PMATRIXLF) - in/out
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the specified rotation should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE
The previous transform is discarded and replaced by the transform describing the specified rotation.

TRANSFORM_ADD
The previous transform is combined with a transform representing the specified rotation in the order (1) previous transform, (2) rotational transform. This option is most useful for incremental updates to transforms.

fxAngle (FIXED) - input
Rotation angle.

The angle describing the rotation, measured counterclockwise from the x-axis in degrees.

pptlCenter (PPOINTL) - input
Center of rotation.

The point about which the rotation occurs.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiRotate - Remarks

This function is a helper function that either applies a specified rotational component to an existing transform matrix, or replaces the matrix with one that represents the specified rotation alone.

The transform is specified as a one-dimensional array of 9 elements that are the elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is as follows:

```
Matrix                  Array


a  b  0
c  d  0           (a,b,0,c,d,0,e,f,1)
e  f  1
```

Transforms act on the coordinates of primitives, so that a point with coordinates (x,y) is transformed to the point:

```
(a*x + c*y + e, b*x + d*y + f)
```

The transform can be used in any call following:

- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix
- GpiSetDefaultViewMatrix.

Other similar helper functions are:

- GpiTranslate to apply a translation component
- GpiScale to apply a scaling component.

GpiRotate - Errors

Possible returns from WinGetLastError

PMERR_INV_TRANSFORM_TYPE (0x20D0)
An invalid options parameter was specified with a transform matrix function.

GpiRotate - Related Functions

Related Functions

- GpiScale
- GpiSetDefaultViewMatrix
- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix
- GpiTranslate

GpiRotate - Example Code

In this example, the viewing transform matrix is rotated 10 degrees counterclockwise from the x-axis. Hence, everything will appear rotated.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>


HPS hps;               /* presentation space handle */
MATRIXLF matlf;        /* transform matrix.         */
POINTL ptlCenter;      /* center of rotation.       */


GpiQueryViewingTransformMatrix(hps,
                               1L,
                               &matlf);
ptlCenter.x = 50L;
ptlCenter.y = 50L;

GpiRotate(hps,
          &matlf,
          TRANSFORM_REPLACE,
          MAKEFIXED(10,0), /* rotate 10 degrees left. the angle    */
                           /* must be passed in fixed format.      */
                           /* see the pmgpi.h file for a           */
                           /* description of the MAKEFIXED macro.  */
          &ptlCenter);
```

GpiRotate - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSaveMetaFile

GpiSaveMetaFile - Syntax

This function saves a metafile in a disk file.

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HMF     hmf;          /*  Metafile handle. */
PSZ     pszFilename;  /*  The name of the file to which the metafile is to be saved. */
BOOL    rc;           /*  Success indicator. */

rc = GpiSaveMetaFile(hmf, pszFilename);
```

GpiSaveMetaFile Parameter - hmf

hmf (HMF) - input
Metafile handle.

GpiSaveMetaFile Parameter - pszFilename

pszFilename (PSZ) - input
The name of the file to which the metafile is to be saved.

This name must be a valid external name. It is an error if a file of this name exists already.

GpiSaveMetaFile Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSaveMetaFile - Parameters

hmf (HMF) - input
Metafile handle.

pszFilename (PSZ) - input
The name of the file to which the metafile is to be saved.

This name must be a valid external name. It is an error if a file of this name exists already.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSaveMetaFile - Remarks

The metafile is deleted from storage; this means that the metafile handle is no longer valid.

The metafile may be reaccessed by GpiLoadMetaFile.

GpiSaveMetaFile - Errors

Possible returns from WinGetLastError

PMERR_INV_HMF (0x207E)
An invalid metafile handle was specified.

PMERR_DOSOPEN_FAILURE (0x2024)
A DosOpen call made during GpiLoadMetaFile or GpiSaveMetaFile gave a good return code but the file was not opened successfully.

PMERR_INSUFFICIENT_DISK_SPACE (0x203D)
The operation terminated through insufficient disk space.

PMERR_METAFILE_IN_USE (0x20D9)
An attempt has been made to access a metafile that is in use by another thread.

PMERR_TOO_MANY_METAFILES_IN_USE (0x2106)
The maximum number of metafiles allowed for a given process was exceeded.

GpiSaveMetaFile - Related Functions

Related Functions

- GpiCopyMetaFile
- GpiDeleteMetaFile
- GpiLoadMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength

- GpiSetMetaFileBits

GpiSaveMetaFile - Example Code

This function saves a metafile in a disk file.

```
#define INCL_GPIMETAFILES
#include <OS2.H>

HMF hmf;    /* metafile handle. */
GpiSaveMetaFile(hmf, "file.met");
```

GpiSaveMetaFile - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSavePS

GpiSavePS - Syntax

This function saves information about the presentation space on a LIFO (last in, first out) stack.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lPSid;  /*  Identifier of saved presentation space. */

lPSid = GpiSavePS(hps);
```

GpiSavePS Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSavePS Return Value - lPSid

lPSid (LONG) - returns
Identifier of saved presentation space.

This may be used on a subsequent GpiRestorePS call. The identifier is equal to the depth of the saved presentation space on the save/restore stack, with 1 representing the base level:

>=1
Identifier of saved presentation space
GPI_ERROR
Error.

GpiSavePS - Parameters

hps (HPS) - input
Presentation-space handle.

lPSid (LONG) - returns
Identifier of saved presentation space.

This may be used on a subsequent GpiRestorePS call. The identifier is equal to the depth of the saved presentation space on the save/restore stack, with 1 representing the base level:

>=1
Identifier of saved presentation space
GPI_ERROR
Error.

GpiSavePS - Remarks

The stack is different from the one used to save attribute values (see GpiSetAttrMode) in a normal presentation space.

This function, and GpiRestorePS, can be used with a micro presentation space, as well as a normal presentation space (in **draw** drawing mode only).

The presentation space itself is unchanged.

The following are saved:

- Current attributes
- Current transforms, viewing limits, and clip path
- Current position
- Reference to selected clip region
- Any loaded logical color table
- References to any loaded logical fonts

- References to the regions created on the associated device context.

The following are not saved:

- Draw controls
- Drawing mode
- Edit mode and attribute mode
- The visible region.

Note: Only references to resources, rather than the actual resources (such as clip region, logical fonts, and regions) are copied by this function, so the actual resources must not be changed.

This function is valid in an open segment bracket, but only if the drawing mode (see GpiSetDrawingMode) is set to **draw**. This function can occur within an open element bracket. When it occurs within an open area or path bracket, GpiRestorePS must be called before the bracket is closed.

If this function occurs during the generation of a metafile, the drawing mode must be set to **draw** when the metafile is replayed.

It is recommended that GpiSavePS and GpiRestorePS are used in pairs and are NOT split across page boundaries in a multiple-page print job.

GpiSavePS - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_DRAW_MODE (0x20DE)
An attempt was made to issue GpiSavePS or GpiRestorePS while the drawing mode was not set to DM_DRAW.

GpiSavePS - Related Functions

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSetPS

GpiSavePS - Example Code

This example uses the GpiSavePS function to save the state of the presentation space. The identifier returned by the function is used in the call to the GpiRestorePS function to restore the saved state.

```
#define INCL_GPICONTROL
#include <OS2.H>
```

```
HPS hps;    /* presentation-space handle. */
LONG idPS;

idPS = GpiSavePS(hps);   /* saves the presentation-space state */
     .
     .
     .
/* restores the presentation-space state */
GpiRestorePS(hps, idPS);
```

GpiSavePS - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiScale

GpiScale - Syntax

This function applies a scaling component to a transform matrix.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
PMATRIXLF    pmatlfArray;  /*  Transform matrix. */
LONG         lOptions;     /*  Transform options. */
PFIXED       afxScale;     /*  Array of scale factors. */
PPOINTL      pptlCenter;   /*  Center of scale. */
BOOL         rc;           /*  Success indicator. */

rc = GpiScale(hps, pmatlfArray, lOptions,
       afxScale, pptlCenter);
```

GpiScale Parameter - hps

hps (HPS) - input

Presentation-space handle.

GpiScale Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - in/out
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

GpiScale Parameter - lOptions

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the specified scaling should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE
The previous transform is discarded and replaced by the transform describing the specified scaling.

TRANSFORM_ADD
The previous transform is combined with a transform representing the specified scaling in the order (1) previous transform, (2) scaling transform. This option is most useful for incremental updates to transforms.

GpiScale Parameter - afxScale

afxScale (PFIXED) - input
Array of scale factors.

The first element of the array is the x scale factor, and the second is the y scale factor.

Scaling values outside the range -1 through +1 are not valid for subsequent use with presentation spaces that have a coordinate format (as set by the GpiCreatePS function) of GPIF_SHORT.

GpiScale Parameter - pptlCenter

pptlCenter (PPOINTL) - input
Center of scale.

The point about which the scale occurs.

GpiScale Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiScale - Parameters

hps (HPS) - input
Presentation-space handle.

pmatlfArray (PMATRIXLF) - in/out
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the specified scaling should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE
The previous transform is discarded and replaced by the transform describing the specified scaling.

TRANSFORM_ADD
The previous transform is combined with a transform representing the specified scaling in the order (1) previous transform, (2) scaling transform. This option is most useful for incremental updates to transforms.

afxScale (PFIXED) - input
Array of scale factors.

The first element of the array is the x scale factor, and the second is the y scale factor.

Scaling values outside the range -1 through +1 are not valid for subsequent use with presentation spaces that have a coordinate format (as set by the GpiCreatePS function) of GPIF_SHORT.

pptlCenter (PPOINTL) - input
Center of scale.

The point about which the scale occurs.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiScale - Remarks

This function is a helper function which either applies a specified scaling component to an existing transform matrix, or replaces the matrix with one that represents the specified scaling alone.

The transform is specified as a one-dimensional array of 9 elements that are the elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

```
Matrix                  Array


a  b  0
c  d  0           (a,b,0,c,d,0,e,f,1)
e  f  1
```

Transforms act on the coordinates of primitives, so that a point with coordinates (x,y) is transformed to the point:

```
(a*x + c*y + e, b*x + d*y + f)
```

The transform can be used in any call following:

- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix
- GpiSetDefaultViewMatrix.

Other similar helper functions are:

- GpiTranslate to apply a translation component
- GpiRotate to apply a rotation component.

GpiScale - Errors

Possible returns from WinGetLastError

PMERR_INV_TRANSFORM_TYPE (0x20D0)
An invalid options parameter was specified with a transform matrix function.

GpiScale - Related Functions

Related Functions

- GpiRotate
- GpiSetDefaultViewMatrix
- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix
- GpiTranslate

GpiScale - Example Code

In this example, the viewing transform matrix is scaled by a factor of 2.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;               /* presentation space handle */
MATRIXLF matlf;        /* transform matrix          */
POINTL ptlCenter;      /* center of rotation        */
FIXED  scalars[2];              /* x and y scale factors     */


GpiQueryViewingTransformMatrix(hps,
                               1L,
                               &matlf);
ptlCenter.x = 50L;
ptlCenter.y = 50L;
scalars[0] =  MAKEFIXED(2,0);   /* Scale x values by 2 */
scalars[1] =  MAKEFIXED(3,0);   /* Scale y values by 3 */

GpiScale (hps,
          &matlf,
          TRANSFORM_REPLACE,
          scalars,
          &ptlCenter);
```

GpiScale - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSelectPalette

GpiSelectPalette - Syntax

This function selects a palette into a presentation space.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>
```

```
HPS     hps;      /*  Presentation-space handle. */
HPAL    hpal;     /*  Palette handle. */
HPAL    hpalOld;  /*  Old palette handle. */

hpalOld = GpiSelectPalette(hps, hpal);
```

GpiSelectPalette Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSelectPalette Parameter - hpal

hpal (HPAL) - input
Palette handle.

NULLHANDLE
Set the color table for the presentation space to the default table (see GpiCreateLogColorTable).

Other
Palette handle.

GpiSelectPalette Return Value - hpalOld

hpalOld (HPAL) - returns
Old palette handle.

NULLHANDLE
Successful completion, default (or loaded) color table was in effect

PAL_ERROR
Error occurred

Otherwise
Old palette handle.

GpiSelectPalette - Parameters

hps (HPS) - input
Presentation-space handle.

hpal (HPAL) - input
Palette handle.

NULLHANDLE

Set the color table for the presentation space to the default table (see GpiCreateLogColorTable).

Other
Palette handle.

hpalOld (HPAL) - returns
Old palette handle.

NULLHANDLE
Successful completion, default (or loaded) color table was in effect
PAL_ERROR
Error occurred
Otherwise
Old palette handle.

GpiSelectPalette - Remarks

This function overrides any color table previously loaded (see GpiCreateLogColorTable), or palette previously selected into this presentation space.

If *hpal* is specified as NULLHANDLE, then the color table for this presentation space is set to the default color table.

Palettes can be selected into more than one presentation space at a time, but only one palette can be selected into a given presentation space at any time.

If a palette is selected into a presentation space that is associated with a device context of type OD_MEMORY (see "DevOpenDC" in the *Presentation Manager Programming Reference*) irreversible changes take place to any bit map selected into the device context.

If a palette is selected into a presentation space that is associated with a device context of type OD_METAFILE or OD_METAFILE_NOQUERY the palette must apply to the entire picture, and must still be selected at the (last) time the metafile device context is dissociated from the presentation space.

GpiSelectPalette - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HPAL (0x2111)
An invalid color palette handle was specified.

PMERR_INSUFFICIENT_MEMORY (0x203E)
The operation terminated through insufficient memory.

PMERR_PALETTE_BUSY (0x2112)
An attempt has been made to reset the owner of a palette when it was busy.

PMERR_INV_IN_AREA (0x2085)
An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

GpiSelectPalette - Related Functions

Related Functions

- GpiAnimatePalette
- GpiCreateLogColorTable
- GpiCreatePalette
- GpiDeletePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSetPaletteEntries

GpiSelectPalette - Example Code

This function selects a palette into a presentation space.

```
#define INCL_GPILOGCOLORTABLE
#include <OS2.H>

HPS hps;       /* presentation-space handle. */
HPAL hpalOld, hpal;  /* old palette handle.        */
hpalOld = GpiSelectPalette(hps, hpal);
```

GpiSelectPalette - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetArcParams

GpiSetArcParams - Syntax

This function sets the current arc parameters.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS           hps;            /*  Presentation-space handle. */
```

```
PARCPARAMS    parcpArcParams;  /*  Arc parameters. */
BOOL          rc;              /*  Success indicator. */

rc = GpiSetArcParams(hps, parcpArcParams);
```

GpiSetArcParams Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetArcParams Parameter - parcpArcParams

parcpArcParams (PARCPARAMS) - input
Arc parameters.

This structure has four elements p, q, r, and s.

GpiSetArcParams Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetArcParams - Parameters

hps (HPS) - input
Presentation-space handle.

parcpArcParams (PARCPARAMS) - input
Arc parameters.

This structure has four elements p, q, r, and s.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE

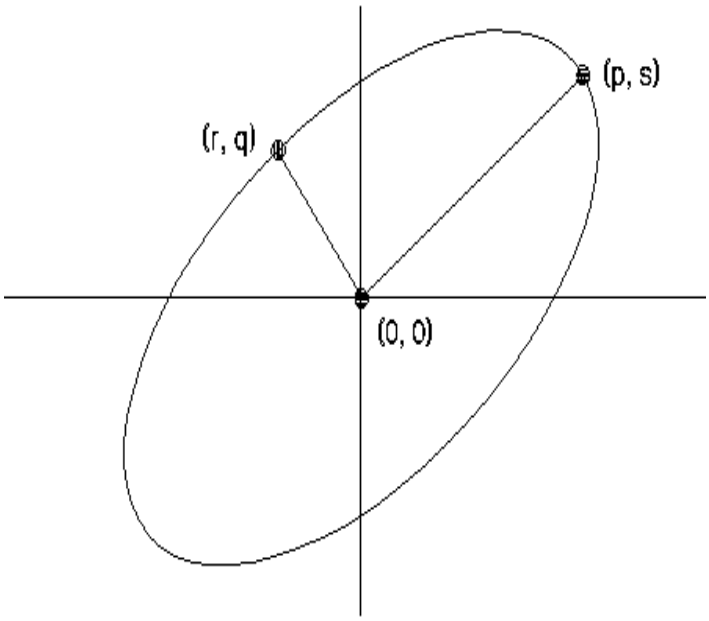
Error occurred.

GpiSetArcParams - Remarks

The arc parameters p, q, r, and s, define the shape and orientation of a ellipse that is used for subsequent GpiPointArc, GpiFullArc, and GpiPartialArc functions. For GpiFullArc and GpiPartialArc, they also determine the direction of drawing, as follows:

- If p*q > r*s the direction is counterclockwise
- If p*q < r*s the direction is clockwise
- If p*q = r*s a straight line is drawn.

For GpiFullArc and GpiPartialArc, these parameters also define the nominal size of the ellipse; this may be changed by using the multiplier.



For GpiPointArc, the size of the ellipse is determined by the three points specified on GpiPointArc.

The arc parameters define a transformation that maps the *unit circle* to the required ellipse, placed at the origin (0,0):

```
x' = p*x + r*y
y' = s*x + q*y
```

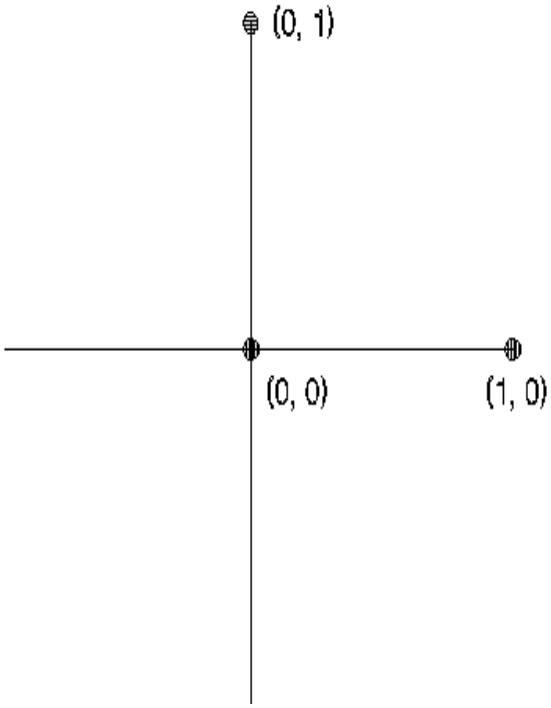
With reference to the figure above, if p*r + s*q = 0, the transform is termed *orthogonal*, and the line from the origin (0,0) to the point (p,s) is either the radius of the circle, or half the major axis of the ellipse. The line from the origin to the point (r,q) is either the radius of the circle, or half of the minor axis of the ellipse.

For maximum accuracy, orthogonal transforms must be used. The matrix must not be singular.

The initial default values of arc parameters (unless changed with GpiSetDefArcParams) are:

```
p = 1     r = 0
s = 0     q = 1
```

producing a unit circle. (See the figure below.).



Arc parameter transformation takes place in world coordinates. Any other non-square transformations in force change the shape of the figure accordingly.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the arc parameters is preserved.

GpiSetArcParams - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetArcParams - Related Functions

Related Functions

- GpiFullArc
- GpiPartialArc
- GpiPointArc
- GpiQueryArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor

- GpiSetDefArcParams
- GpiSetDefAttrs
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix

GpiSetArcParams - Graphic Elements and Orders

Element Type: **OCODE_GSAP**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Arc Parameters

Element Type: **OCODE_GPSAP**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Set Arc Parameters

GpiSetArcParams - Example Code

This example uses the GpiSetArcParams function to draw an ellipse. The semimajor axis of the ellipse is 100, and the semiminor axis is 50. These values are in world coordinates, computed using the lP and lQ values of the arc parameters and the multiplier provided with the GpiFullArc function.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;                          /*  Presentation-space */
                                  /*  handle.            */
ARCPARAMS arcp = { 4, 2, 0, 0 };  /* Arc parameters.          */
                                  /* This structure has four  */
                                  /* elements p, q, r, and s. */

POINTL ptl = {100, 100};
GpiSetArcParams(hps, &arcp);
GpiMove(hps, &ptl);
GpiFullArc(hps, DRO_OUTLINE, MAKEFIXED(25, 0));
```

GpiSetArcParams - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetAttrMode

GpiSetAttrMode - Syntax

This function specifies the current attribute mode.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lMode;  /*  Attribute mode. */
BOOL    rc;     /*  Success indicator. */

rc = GpiSetAttrMode(hps, lMode);
```

GpiSetAttrMode Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetAttrMode Parameter - lMode

lMode (LONG) - input
Attribute mode.

AM_PRESERVE
Preserve attributes
AM_NOPRESERVE
Do not preserve attributes.

GpiSetAttrMode Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
: Error occurred.

GpiSetAttrMode - Parameters

hps (HPS) - input
: Presentation-space handle.

lMode (LONG) - input
: Attribute mode.

AM_PRESERVE
: Preserve attributes

AM_NOPRESERVE
: Do not preserve attributes.

rc (BOOL) - returns
: Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetAttrMode - Remarks

The attribute mode is used to specify whether a primitive attribute is to be preserved when set to a new value by a subsequent attribute setting call. The preserved value of an attribute can be restored using the GpiPop function. Any attributes that have been preserved in a called segment are automatically restored on return to the caller. The values of any attributes preserved in a *chained* segment, however, that are not restored using GpiPop by the end of the segment, are lost.

The following are affected:

- GpiSetArcParams
- GpiSetBackColor (applies individually by primitive type if GpiSetAttrs is used)
- GpiSetBackMix (applies individually by primitive type if GpiSetAttrs is used)
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor (applies individually by primitive type if GpiSetAttrs is used)
- GpiSetCurrentPosition
- GpiSetMix (applies individually by primitive type if GpiSetAttrs is used)
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetMarker
- GpiSetModelTransformMatrix
- GpiSetPattern
- GpiSetPatternRefPoint
- GpiSetPatternSet
- GpiSetTag.

The initial value of the attribute mode, that is, its value before this function is issued, is AM_NOPRESERVE.

Attribute mode applies to attributes passed across the API using GpiSet... calls. What mode to use for a particular GpiSet... call is decided by the attribute mode current at the time the GpiSet... call is passed across the API. The mode may be changed at any time, and does not affect any attribute setting calls that have already been retained in the segment store.

Attribute mode only applies to individual GpiSet... calls (including GpiSetAttrs and calls such as GpiSetColor). It does not apply to any attribute setting calls passed across in bulk, such as GpiPutData, GpiElement, and GpiPlayMetaFile; these already indicate individually whether they should cause the attribute to be preserved.

Attribute mode cannot be set for GPIT_MICRO type presentation spaces, or for presentation spaces associated with OD_METAFILE_NOQUERY type device contexts (see GpiCreatePS and DevOpenDC). In these cases the presentation space behaves as if AM_NOPRESERVE is in operation.

GpiSetAttrMode - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_ATTR_MODE (0x2043)
: An invalid mode parameter was specified with GpiSetAttrMode.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_DC_TYPE (0x2060)
: An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.

GpiSetAttrMode - Related Functions

Related Functions

- GpiPop
- GpiQueryAttrMode
- GpiQueryAttrs
- GpiResetPS
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetPS

GpiSetAttrMode - Example Code

This example uses the GpiSetAttrMode function to set the attribute mode to preserve.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;                        /*  Presentation-space */
                                /*  handle.            */
```

```
POINTL ptl[2] = { 50, 50, 100, 100 };

 GpiSetColor(hps, CLR_BLUE);
 GpiSetAttrMode(hps, AM_PRESERVE);/* sets attribute mode to */
                                  /* preserve.                  */
 GpiSetColor(hps, CLR_GREEN);     /* changes color and saves old */
                                  /* color.                     */
 GpiLine(hps, &ptl[0]);           /* draws green line           */
 GpiPop(hps, 1L);                 /* pops old attribute from    */
                                  /* stack.                     */
 GpiLine(hps, &ptl[1]);           /* draws blue line            */
```

GpiSetAttrMode - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetAttrs

GpiSetAttrs - Syntax

This function sets attributes for the specified primitive type.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lPrimType;   /*  Primitive type. */
ULONG      flAttrMask;  /*  Attributes mask. */
ULONG      flDefMask;   /*  Defaults mask. */
PBUNDLE    ppbunAttrs;  /*  Attributes. */
BOOL       rc;          /*  Success indicator. */

rc = GpiSetAttrs(hps, lPrimType, flAttrMask,
       flDefMask, ppbunAttrs);
```

GpiSetAttrs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetAttrs Parameter - lPrimType

lPrimType (LONG) - input
Primitive type.

The primitive type for which attributes are to be set:

PRIM_LINE	Line and arc primitives
PRIM_CHAR	Character primitives
PRIM_MARKER	Marker primitives
PRIM_AREA	Area primitives
PRIM_IMAGE	Image primitives.

GpiSetAttrs Parameter - flAttrMask

flAttrMask (ULONG) - input
Attributes mask.

Each flag set indicates that either the corresponding flag in *flDefMask* is set, or the *ppbunAttrs* buffer contains data for the corresponding attribute. If all of the flags in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

Line attributes:

LBB_COLOR	Line color
BB_BACK_COLOR	Line background color
LBB_MIX_MODE	Line mix
LBB_BACK_MIX_MODE	Line background mix
LBB_WIDTH	Line width
LBB_GEOM_WIDTH	Geometric line width
LBB_TYPE	Line type
LBB_END	Line end
LBB_JOIN	Line join.

Character attributes:

CBB_COLOR	Character color
CBB_BACK_COLOR	Character background color
CBB_MIX_MODE	Character mix
CBB_BACK_MIX_MODE	Character background mix
CBB_SET	Character set
CBB_MODE	Character mode
CBB_BOX	Character box
CBB_ANGLE	Character angle
CBB_SHEAR	Character shear
CBB_DIRECTION	Character direction
CBB_TEXT_ALIGN	Character text alignment
CBB_EXTRA	Character extra
CBB_BREAK_EXTRA	Character break extra

Marker attributes:

MBB_COLOR	Marker color
MBB_BACK_COLOR	Marker background color
MBB_MIX_MODE	Marker mix
MBB_BACK_MIX_MODE	Marker background mix
MBB_SET	Marker set
MBB_SYMBOL	Marker symbol
MBB_BOX	Marker box.

Pattern attributes (areas):

ABB_COLOR	Area color
ABB_BACK_COLOR	Area background color
ABB_MIX_MODE	Area mix
ABB_BACK_MIX_MODE	Area background mix
ABB_SET	Pattern set
ABB_SYMBOL	Pattern symbol
ABB_REF_POINT	Pattern reference point.

Image attributes:

IBB_COLOR	Image color
IBB_BACK_COLOR	Image background color
IBB_MIX_MODE	Image mix
IBB_BACK_MIX_MODE	Image background mix.

GpiSetAttrs Parameter - flDefMask

flDefMask (ULONG) - input
Defaults mask.

Each flag set (and for which *flAttrMask* is also set) causes the corresponding attribute to be set to its default value.

GpiSetAttrs Parameter - ppbunAttrs

ppbunAttrs (PBUNDLE) - input
Attributes.

This is a structure containing the attribute value of each attribute for which the *flAttrMask* flag is set (and which is not to be set to its default value), at the correct offset as specified below for the particular primitive type.

Primitive type	Structure
Line attributes	LINEBUNDLE
Character attributes	CHARBUNDLE
Marker attributes	MARKERBUNDLE
Pattern attributes (areas)	AREABUNDLE
Image attributes	IMAGEBUNDLE

GpiSetAttrs Return Value - rc

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetAttrs - Parameters

hps (HPS) - input
Presentation-space handle.

lPrimType (LONG) - input
Primitive type.

The primitive type for which attributes are to be set:

PRIM_LINE
: Line and arc primitives

PRIM_CHAR
: Character primitives

PRIM_MARKER
: Marker primitives

PRIM_AREA
: Area primitives

PRIM_IMAGE
: Image primitives.

flAttrMask (ULONG) - input
Attributes mask.

Each flag set indicates that either the corresponding flag in *flDefMask* is set, or the *ppbunAttrs* buffer contains data for the corresponding attribute. If all of the flags in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

Line attributes:

LBB_COLOR
: Line color

BB_BACK_COLOR
: Line background color

LBB_MIX_MODE
: Line mix

LBB_BACK_MIX_MODE
: Line background mix

LBB_WIDTH
: Line width

LBB_GEOM_WIDTH
: Geometric line width

LBB_TYPE
: Line type

LBB_END
: Line end

LBB_JOIN
: Line join.

Character attributes:

CBB_COLOR
: Character color

CBB_BACK_COLOR
: Character background color

CBB_MIX_MODE
: Character mix

CBB_BACK_MIX_MODE
: Character background mix

CBB_SET
: Character set

CBB_MODE
: Character mode

CBB_BOX
: Character box

CBB_ANGLE
: Character angle

CBB_SHEAR
: Character shear

CBB_DIRECTION
: Character direction

CBB_TEXT_ALIGN
: Character text alignment

CBB_EXTRA
: Character extra

CBB_BREAK_EXTRA
: Character break extra

Marker attributes:

MBB_COLOR
: Marker color

MBB_BACK_COLOR	Marker background color
MBB_MIX_MODE	Marker mix
MBB_BACK_MIX_MODE	Marker background mix
MBB_SET	Marker set
MBB_SYMBOL	Marker symbol
MBB_BOX	Marker box.

Pattern attributes (areas):

ABB_COLOR	Area color
ABB_BACK_COLOR	Area background color
ABB_MIX_MODE	Area mix
ABB_BACK_MIX_MODE	Area background mix
ABB_SET	Pattern set
ABB_SYMBOL	Pattern symbol
ABB_REF_POINT	Pattern reference point.

Image attributes:

IBB_COLOR	Image color
IBB_BACK_COLOR	Image background color
IBB_MIX_MODE	Image mix
IBB_BACK_MIX_MODE	Image background mix.

flDefMask (ULONG) - input
Defaults mask.

Each flag set (and for which *flAttrMask* is also set) causes the corresponding attribute to be set to its default value.

ppbunAttrs (PBUNDLE) - input
Attributes.

This is a structure containing the attribute value of each attribute for which the *flAttrMask* flag is set (and which is not to be set to its default value), at the correct offset as specified below for the particular primitive type.

Primitive type	Structure
Line attributes	LINEBUNDLE
Character attributes	CHARBUNDLE
Marker attributes	MARKERBUNDLE
Pattern attributes (areas)	AREABUNDLE
Image attributes	IMAGEBUNDLE

rc (BOOL) - returns
Success indicator.

TRUE	Successful completion
FALSE	Error occurred.

GpiSetAttrs - Remarks

Any attribute (for the specified primitive type) for which the appropriate flag is set in the *flAttrMask* has its value updated:

- If the corresponding flag in *flDefMask* is also set, the attribute is set to default.
- If the corresponding flag in *flDefMask* is not set, the attribute is set to the value specified in the *ppbunAttrs* structure.

Any attribute for which the appropriate flag in *flAttrMask* is not set is unchanged, regardless of the setting of the corresponding flag in *flDefMask* .

The *flDefMask* and *flAttrMask* parameters each contain flags: each attribute of the primitive type in question is represented by one flag.

The data in the *ppbunAttrs* buffer consists of a structure of attribute data. The layout of the structure is fixed for each primitive type. Only data for attributes for which the flag is set in *flAttrMask* (but not in *flDefMask*) is inspected; any other data is ignored.

Note: The buffer need be no longer than is necessary to contain the data for the highest offset attribute referenced.

If default values of attributes are required, they must be requested using the *flDefMask* . The system does not recognize attribute values whose meaning would normally be "use default" in the *ppbunAttrs* buffer.

Where possible, invalid color values are detected by this call and cause an error (PMERR_INV_COLOR_ATTR or PMERR_INV_BACKGROUND_COL_ATTR) to be logged. Some invalid color values cannot be detected until draw time at which point the implementation optionally defaults them, or causes the above error to be logged. If an attempt to set an invalid value by this function is detected, none of the specified attributes is changed. Note, however, that some invalid attribute values (for example, colors and mixes) may not be detected until the attribute is used.

If this function occurs within a path bracket, it must only set:

- Line attributes, other than the geometric line width
- Character attributes, other than foreground or background colors or mixes
- Marker attributes, other than foreground or background colors or mixes.

The default values of attributes can be changed with GpiSetDefAttrs.

The attribute mode (see GpiSetAttrMode) determines whether the current values of the attributes are preserved. If they are, one "push" call is generated for each affected attribute, in the order in which the attributes are specified, in the appropriate xxxBUNDLE structure.

GpiSetAttrs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PRIMITIVE_TYPE (0x20B9)
An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.

PMERR_UNSUPPORTED_ATTR (0x2109)
An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.

PMERR_INV_COLOR_ATTR (0x2053)
An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_BACKGROUND_COL_ATTR (0x2044)
An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_MIX_ATTR (0x20A3)
An invalid mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_LINE_WIDTH_ATTR (0x2096)
An invalid line width attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_GEOM_LINE_WIDTH_ATTR (0x2078)
An invalid geometric line width attribute value was specified.

PMERR_INV_LINE_TYPE_ATTR (0x2095)
An invalid line type attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_LINE_END_ATTR (0x2093)
An invalid line end attribute value was specified.

PMERR_INV_LINE_JOIN_ATTR (0x2094)
An invalid line join attribute value was specified.

PMERR_INV_CHAR_SET_ATTR (0x204F)

An invalid character setid attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_CHAR_MODE_ATTR (0x204D)
An invalid character mode attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_CHAR_DIRECTION_ATTR (0x204C)
An invalid character direction attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_CHAR_SHEAR_ATTR (0x2050)
An invalid character shear attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_CHAR_ANGLE_ATTR (0x204B)
The default character angle attribute value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_MARKER_SET_ATTR (0x2099)
An invalid marker set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_MARKER_SYMBOL_ATTR (0x209A)
An invalid marker symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_PATTERN_SET_ATTR (0x20B2)
An invalid pattern set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_PATTERN_ATTR (0x20B0)
An invalid pattern symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_UNSUPPORTED_ATTR_VALUE (0x210A)
An attribute value was specified with GpiSetAttrs that is not supported.

PMERR_INV_PATTERN_SET_FONT (0x20B3)
An attempt was made to use an unsuitable font as a pattern set.

PMERR_HUGE_FONTS_NOT_SUPPORTED (0x2035)
An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

GpiSetAttrs - Related Functions

Related Functions

- GpiPop
- GpiQueryAttrs
- GpiSetAttrMode
- GpiSetDefAttrs

GpiSetAttrs - Graphic Elements and Orders

The element type depends on the *lPrimType* parameter.

For each element, the "Set" orders are generated, if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE, and the "Push and Set" orders if it is AM_PRESERVE. In either instance, a particular order is generated only if the corresponding attribute is being set with this function, as specified on the *ppbunAttrs* parameter.

Element Type: **ETYPE_LINEBUNDLE**
Generated if *lPrimType* is PRIM_LINE.

Order: Set Individual Attribute
One of these for each of LBB_COLOR and LBB_BACK_MIX_MODE, as required.

Order: Set Fractional Line Width
LBB_WIDTH, as required.

Order: Set Stroke Line Width
LBB_GEOM_WIDTH, as required.

Order: Set Line Type
LBB_TYPE, as required.

Order: Set Line End
LBB_END, as required.

Order: Set Line Join
LBB_JOIN, as required.

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: Push and Set Individual Attribute
One of these for each of LBB_COLOR and LBB_MIX_MODE, as required.

Order: Push and Set Fractional Line Width
LBB_WIDTH, as required.

Order: Push and Set Stroke Line Width
LBB_GEOM_WIDTH, as required.

Order: Push and Set Line Type
LBB_TYPE, as required.

Order: Push and Set Line End
LBB_END, as required.

Order: Push and Set Line Join
LBB_JOIN, as required.

Element Type: **ETYPE_CHARBUNDLE**
Generated if *lPrimType* is PRIM_CHAR.

Order: Set Individual Attribute
One of these for each of CBB_COLOR, CBB_BACK_COLOR, CBB_MIX_MODE, and CBB_BACK_MIX_MODE, as required.

Order: Set Character Set
CBB_SET

Order: Set Character Precision
CBB_MODE

Order: Set Character Cell
CBB_BOX

Order: Set Character Angle
CBB_ANGLE

Order: Set Character Shear
CBB_SHEAR

Order: Set Character Direction
CBB_DIRECTION

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: Push and Set Individual Attribute

One of these for each of CBB_COLOR, CBB_BACK_COLOR, CBB_MIX_MODE, and CBB_BACK_MIX_MODE, as required.

Order: Push and Set Character Set
CBB_SET

Order: Push and Set Character Precision
CBB_MODE

Order: Set Character Cell
CBB_BOX

Order: Set Character Angle
CBB_ANGLE

Order: Push and Set Character Shear
CBB_SHEAR

Order: Push and Set Character Direction
CBB_DIRECTION

Element Type: **ETYPE_MARKERBUNDLE**
Generated if *IPrimType* is PRIM_MARKER.

Order: Set Individual Attribute
One of these for each of MBB_COLOR, MBB_BACK_COLOR, MBB_MIX_MODE, and MBB_BACK_MIX_MODE, as required.

Order: Set Marker Set
MBB_SET

Order: Set Marker Symbol
MBB_SYMBOL

Order: Set Marker Cell
MBB_BOX

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: Push and Set Individual Attribute
One of these for each of MBB_COLOR, MBB_BACK_COLOR, MBB_MIX_MODE, and MBB_BACK_MIX_MODE, as required.

Order: Push and Set Marker Set
MBB_SET

Order: Push and Set Marker Symbol
MBB_SYMBOL

Order: Push and Set Marker Cell
MBB_BOX

Element Type: **ETYPE_AREABUNDLE**
Generated if *IPrimType* is PRIM_AREA.

Order: Set Individual Attribute
One of these for each of ABB_COLOR, ABB_BACK_COLOR, ABB_MIX_MODE, and ABB_BACK_MIX_MODE, as required.

Order: Set Pattern Set
ABB_SET

Order: Set Pattern Symbol
ABB_SYMBOL

Order: Set Pattern Reference Point
ABB_REF_POINT

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: Push and Set Individual Attribute
One of these for each of ABB_COLOR, ABB_BACK_COLOR, ABB_MIX_MODE, and ABB_BACK_MIX_MODE, as required.

Order: Push and Set Pattern Set
ABB_SET

Order: Push and Set Pattern Symbol
ABB_SYMBOL

Order: Push and Set Pattern Reference Point
ABB_REF_POINT

Element Type: **ETYPE_IMAGEBUNDLE**
Generated if *lPrimType* is PRIM_IMAGE.

Order: Set Individual Attribute
One of these for each of IBB_COLOR, IBB_BACK_COLOR, IBB_MIX_MODE, and IBB_BACK_MIX_MODE, as required.

As many as required of the following are generated if the attribute mode is AM_PRESERVE:

Order: Push and Set Individual Attribute
One of these for each of IBB_COLOR, IBB_BACK_COLOR, IBB_MIX_MODE, and IBB_BACK_MIX_MODE, as required.

GpiSetAttrs - Example Code

This example uses the GpiSetAttrs function to set the line color to red and the line width to its default value.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;                          /*  Presentation-space */
                                  /*  handle.            */

LINEBUNDLE lbnd;
lbnd.lColor = CLR_RED;

 GpiSetAttrs(hps,               /* presentation-space handle  */
     PRIM_LINE,                 /* line primitive.            */
     LBB_COLOR | LBB_WIDTH,     /* sets line color and width. */
     LBB_WIDTH,                 /* sets line width to default */
     &lbnd);                    /* buffer for attributes.     */
```

GpiSetAttrs - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetBackColor

GpiSetBackColor - Syntax

This function sets the current background color index attribute, for each individual primitive type, to the specified value.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lColor;  /*  Background color. */
BOOL    rc;      /*  Success indicator. */

rc = GpiSetBackColor(hps, lColor);
```

GpiSetBackColor Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetBackColor Parameter - lColor

lColor (LONG) - input
Background color.

It can be either an index value, if the color table is in index mode, or an RGB value, if the color table is in RGB mode. For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

CLR_FALSE
All color planes are zeros.

CLR_TRUE
All color planes are ones.

CLR_DEFAULT
Set to default value. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see "WinSetSysColors" in the *Presentation Manager Programming Reference*). For a printer, it is the paper color. The default can be changed by setting new system colors from the control panel for the display, or by selecting a paper color for a printer (if allowed by the device driver), or set explicitly with GpiSetDefAttrs.

CLR_WHITE
White (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to white.

CLR_BLACK
Black (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to black.

CLR_BACKGROUND

Reset color, used by GpiErase. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see "WinSetSysColors" in the *Presentation Manager Programming* Reference). see WinSetSysColors) for the default color table. For a printer, it is the paper color. For a loaded color table, it is color index 0. For an RGB color table, it is color 000000 (black).

CLR_BLUE
Blue (default color table).

CLR_RED
Red (default color table).

CLR_PINK
Pink (default color table).

CLR_GREEN
Green (default color table).

CLR_CYAN
Cyan (default color table).

CLR_YELLOW
Yellow (default color table).

CLR_NEUTRAL
Neutral (default color table). A device-dependent color, that for the default color table provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT: see "WinSetSysColors" in the *Presentation Manager Programming Reference*). For a printer, it is a color that contrasts with the paper color. For a loaded color table, it is color index 7; in RGB mode it is interpreted as color 000007.

CLR_DARKGRAY
Dark gray (default color table).

CLR_DARKBLUE
Dark blue (default color table).

CLR_DARKRED
Dark red (default color table).

CLR_DARKPINK
Dark pink (default color table).

CLR_DARKGREEN
Dark green (default color table).

CLR_DARKCYAN
Dark cyan (default color table).

CLR_BROWN
Brown (default color table).

CLR_PALEGRAY
Pale gray (default color table).

For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

GpiSetBackColor Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetBackColor - Parameters

hps (HPS) - input
Presentation-space handle.

lColor (LONG) - input
Background color.

It can be either an index value, if the color table is in index mode, or an RGB value, if the color table is in RGB mode. For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

CLR_FALSE
All color planes are zeros.

CLR_TRUE
All color planes are ones.

CLR_DEFAULT
Set to default value. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see "WinSetSysColors" in the *Presentation Manager Programming Reference*). For a printer, it is the paper color. The default can be changed by setting new system colors from the control panel for the display, or by selecting a paper color for a printer (if allowed by the device driver), or set explicitly with GpiSetDefAttrs.

CLR_WHITE
White (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to white.

CLR_BLACK
Black (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to black.

CLR_BACKGROUND
Reset color, used by GpiErase. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see "WinSetSysColors" in the *Presentation Manager Programming* Reference). see WinSetSysColors) for the default color table. For a printer, it is the paper color. For a loaded color table, it is color index 0. For an RGB color table, it is color 000000 (black).

CLR_BLUE
Blue (default color table).

CLR_RED
Red (default color table).

CLR_PINK
Pink (default color table).

CLR_GREEN
Green (default color table).

CLR_CYAN
Cyan (default color table).

CLR_YELLOW
Yellow (default color table).

CLR_NEUTRAL
Neutral (default color table). A device-dependent color, that for the default color table provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT: see "WinSetSysColors" in the *Presentation Manager Programming Reference*). For a printer, it is a color that contrasts with the paper color. For a loaded color table, it is color index 7; in RGB mode it is interpreted as color 000007.

CLR_DARKGRAY
Dark gray (default color table).

CLR_DARKBLUE

Dark blue (default color table).

CLR_DARKRED
Dark red (default color table).

CLR_DARKPINK
Dark pink (default color table).

CLR_DARKGREEN
Dark green (default color table).

CLR_DARKCYAN
Dark cyan (default color table).

CLR_BROWN
Brown (default color table).

CLR_PALEGRAY
Pale gray (default color table).

For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetBackColor - Remarks

Note that if the background mix is BM_LEAVEALONE (the default setting), the background color is not seen.

An attempt to set a negative color value, other than one for which a constant is defined, causes the error PMERR_INV_COLOR_ATTR to be logged. Other color values are allowed, although an error is generated when the color value is needed for drawing if it is invalid for the color table in use at that time (see GpiCreateLogColorTable).

For details of how colors are handled on monochrome devices, also see GpiCreateLogColorTable.

The attribute mode determines whether the current value of the background color attribute is preserved. If it is, the values of the background color attribute, for each primitive type, are preserved, and a single GpiPop function restores them.

This function must not be used within a path or area bracket.

GpiSetBackColor - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_BACKGROUND_COL_ATTR (0x2044)
An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetBackColor - Related Functions

Related Functions

- GpiBeginArea
- GpiBox
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiEndArea
- GpiFullArc
- GpiMarker
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolyMarker
- GpiPolySpline
- GpiQueryBackColor
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix
- GpiQueryCharStringPosAt

GpiSetBackColor - Graphic Elements and Orders

Element Type: **OCODE_GSBICOL**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Background Indexed Color

Element Type: **OCODE_GPSBICOL**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Background Indexed Color

GpiSetBackColor - Example Code

This is an example of a function used to repaint the window when a WM_PAINT message is issued.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

void ClientPaint( HWND hwnd )
{
  POINTL  pt;
  HPS     hps;                      /* Presentation space handle */
  RECTL   rcl;                      /* Window rectangle          */
  long    clrText;
```

```
  /* Obtain a cache PS and set color
     and background mix attributes   */
  hps = WinBeginPaint( hwnd, (HPS)NULLHANDLE, (PRECTL)&rcl );
  GpiSetColor( hps, clrText );
  GpiSetBackColor( hps, CLR_BACKGROUND ); /* set background
                                             to white.        */
  GpiSetBackMix( hps, BM_OVERPAINT );
}
```

GpiSetBackColor - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetBackMix

GpiSetBackMix - Syntax

This function sets the current background mix attribute for each individual primitive type.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lMixMode;  /*  Background-mix mode. */
BOOL    rc;        /*  Success indicator. */

rc = GpiSetBackMix(hps, lMixMode);
```

GpiSetBackMix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetBackMix Parameter - lMixMode

lMixMode (LONG) - input
Background-mix mode.

Background mixes marked with an asterisk (*) are mandatory for all devices.

The currently associated device supports any of the mixes specified as supported in DevQueryCaps (CAPS_BACKGROUND_MIX_SUPPORT) Any other valid mixes can be supported for some primitive types but otherwise result in BM_LEAVEALONE. An error is raised only if the value specified is not one of those listed.

For more information on mixing, see GpiSetMix.

BM_DEFAULT
The default value (BM_LEAVEALONE unless changed with GpiSetDefAttrs).

BM_OR
Logical-OR.

BM_OVERPAINT
The background of the primitive takes precedence over whatever is underneath. (*)

BM_XOR
Exclusive-OR.

BM_LEAVEALONE
The background of the primitive has no effect on what is underneath. (*)

GpiSetBackMix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetBackMix - Parameters

hps (HPS) - input
Presentation-space handle.

lMixMode (LONG) - input
Background-mix mode.

Background mixes marked with an asterisk (*) are mandatory for all devices.

The currently associated device supports any of the mixes specified as supported in DevQueryCaps (CAPS_BACKGROUND_MIX_SUPPORT) Any other valid mixes can be supported for some primitive types but otherwise result in BM_LEAVEALONE. An error is raised only if the value specified is not one of those listed.

For more information on mixing, see GpiSetMix.

BM_DEFAULT
: The default value (BM_LEAVEALONE unless changed with GpiSetDefAttrs).

BM_OR
: Logical-OR.

BM_OVERPAINT
: The background of the primitive takes precedence over whatever is underneath. (*)

BM_XOR
: Exclusive-OR.

BM_LEAVEALONE
: The background of the primitive has no effect on what is underneath. (*)

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetBackMix - Remarks

The background mix attribute controls the way that the background color of a primitive is combined with the color of any primitive that it overlaps.

These primitives are affected by the background mix attribute:

Areas	The background of an area is defined to be every pel within the area that is not set by the shading pattern.
Text	The background of a character is the complete character box.
Images	For an image, the background is every pel within the image that is not set.
Markers	The background of a marker is the complete marker box.

Note: When the background mix is BM_LEAVEALONE (initial default) the background color is not seen.

The attribute mode determines whether the current value of the background mix attribute is preserved. If it is, the values of the background mix attribute for each primitive type are preserved, and a single GpiPop function restores them.

This function should not be used within a path or area bracket.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetBackMix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_BACKGROUND_COL_ATTR (0x2044)
An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetBackMix - Related Functions

Related Functions

- GpiBox
- GpiBeginArea
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiEndArea
- GpiFullArc
- GpiMarker
- GpiPolyMarker
- GpiQueryBackMix
- GpiSetBackColor
- GpiSetColor
- GpiSetMix
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs

GpiSetBackMix - Graphic Elements and Orders

Element Type: **OCODE_GSBMX**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Background Mix

Element Type: **OCODE_GPSBMX**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Set Background Mix

GpiSetBackMix - Example Code

This is an example of a function used to repaint the window when a WM_PAINT message is issued.

```
VOID cdecl ClientPaint( HWND hwnd )
{
  POINTL  pt;
  HPS     hps;                    /* Presentation space handle */
  RECTL   rcl;                    /* Window rectangle          */
```

```
  /* Obtain a cache PS and set color
     and background mix attributes   */
  hps = WinBeginPaint( hwnd, (HPS)NULLHANDLE, (PRECTL)&rcl );
  GpiSetColor( hps, clrText );
  GpiSetBackColor( hps, CLR_BACKGROUND ); /* set background
                                             to white.        */
  GpiSetBackMix( hps, BM_OVERPAINT );
                       /* the background of the primitive takes
                          over whatever is underneath.          */
}
```

GpiSetBackMix - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetBitmap

GpiSetBitmap - Syntax

This function sets a bit map as the currently selected bit map in a memory device context.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hps;     /*  Presentation-space handle. */
HBITMAP    hbm;     /*  Handle of the bit map to be set. */
HBITMAP    hbmOld;  /*  Old bit-map handle. */

hbmOld = GpiSetBitmap(hps, hbm);
```

GpiSetBitmap Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetBitmap Parameter - hbm

hbm (HBITMAP) - input
Handle of the bit map to be set.

A null handle causes the currently selected bit map, in the associated device, to be freed.

It is an error if the bit map is currently tagged for area shading (see GpiSetBitmapId).

GpiSetBitmap Return Value - hbmOld

hbmOld (HBITMAP) - returns
Old bit-map handle.

NULLHANDLE
: Correct (null handle)

HBM_ERROR
: Error

Otherwise
: Old bit-map handle.

GpiSetBitmap - Parameters

hps (HPS) - input
Presentation-space handle.

hbm (HBITMAP) - input
Handle of the bit map to be set.

A null handle causes the currently selected bit map, in the associated device, to be freed.

It is an error if the bit map is currently tagged for area shading (see GpiSetBitmapId).

hbmOld (HBITMAP) - returns
Old bit-map handle.

NULLHANDLE
: Correct (null handle)

HBM_ERROR
: Error

Otherwise
: Old bit-map handle.

GpiSetBitmap - Remarks

The specified presentation space must be currently associated with a memory device context. The device context can represent a different

physical device from the one that the bit map originally loaded or created, providing its format is convertible to one supported on the new device. This is ensured when one of the standard formats is being used.

If a bit map is already current in the device context, the handle of this bit map is returned, before the new bit map is selected.

It is an error if the new bit map is already selected as the current bit map in any device context.

GpiSetBitmap - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HBITMAP (0x207B)
: An invalid bit-map handle was specified.

PMERR_BITMAP_IN_USE (0x2008)
: An attempt was made either to set a bit map into a device context using GpiSetBitmap while it was already selected into an existing device context, or to tag a bit map with a local pattern set identifier (setid) using GpiSetBitmapId while it was already tagged with an existing setid.

PMERR_INCOMPATIBLE_BITMAP (0x203A)
: An attempt was made to select a bit map or perform a BitBlt operation on a device context that was incompatible with the format of the bit map.

PMERR_HBITMAP_BUSY (0x2032)
: An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiSetBitmap - Related Functions

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt

GpiSetBitmap - Example Code

This example uses the GpiSetBitmap function to set a newly created bit map in the device context for the associated presentation space.

Once set, the example initializes the bit map image by drawing in the presentation space.

```
#define INCL_GPIBITMAPS
#define INCL_GPIPRIMITIVES
#include <OS2.H>

 HPS     hps;                        /* Presentation space handle */
 BITMAPINFOHEADER2 bmp = {12, 64, 64, 1, 1};/* 64x64 mono bit map */
 HBITMAP hbm, hbmOld;
 POINTL ptlStart = { 0, 0 };
 POINTL aptlTriangle[3] = { 32, 32, 63, 63, 0, 0 };
 POINTL ptl = { 63, 63 };

 hbm = GpiCreateBitmap(hps,
                       &bmp,
                       0L,
                       NULL,
                       NULL);

 /* Set the bit map and draw in it. */

 /* sets bit map in device context */
 hbmOld = GpiSetBitmap(hps, hbm);
 GpiMove(hps, &ptlStart);
 GpiBox(hps, DRO_FILL, &ptl, 0L, 0L); /* fills in the bit map */
 GpiPolyLine(hps,                      /* draws a triangle        */
             1L,
             aptlTriangle);
 GpiSetBitmap(hps, hbmOld);          /* restores the old bit map */
```

GpiSetBitmap - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetBitmapBits

GpiSetBitmapBits - Syntax

This function transfers bit-map data from application storage to a bit map.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS             hps;            /*  Presentation-space handle. */
LONG            lScanStart;     /*  Line number. */
```

```
LONG            lScans;         /*  Number of scan lines to be transmitted. */
PBYTE           pbBuffer;       /*  Bit-map data buffer. */
PBITMAPINFO2    pbmiInfoTable;  /*  Bit-map information table. */
LONG            lScansSet;      /*  Number of scan lines actually set. */

lScansSet = GpiSetBitmapBits(hps, lScanStart,
              lScans, pbBuffer, pbmiInfoTable);
```

GpiSetBitmapBits Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetBitmapBits Parameter - lScanStart

lScanStart (LONG) - input
Line number.

Scan-line number at which the data transfer is to start, counting from 0 as the bottom line.

GpiSetBitmapBits Parameter - lScans

lScans (LONG) - input
Number of scan lines to be transmitted.

It must be greater or equal to 0.

GpiSetBitmapBits Parameter - pbBuffer

pbBuffer (PBYTE) - input
Bit-map data buffer.

Address in application storage from which the bit-map data is to be copied. This field must point to a doubleword aligned address to assure correct creation of the bit map by the device in device specific format.

GpiSetBitmapBits Parameter - pbmiInfoTable

pbmiInfoTable (PBITMAPINFO2) - input
Bit-map information table.

GpiSetBitmapBits Return Value - lScansSet

lScansSet (LONG) - returns
Number of scan lines actually set.

>=0
Number of scan lines actually set
GPI_ALTERROR
Error.

GpiSetBitmapBits - Parameters

hps (HPS) - input
Presentation-space handle.

lScanStart (LONG) - input
Line number.

Scan-line number at which the data transfer is to start, counting from 0 as the bottom line.

lScans (LONG) - input
Number of scan lines to be transmitted.

It must be greater or equal to 0.

pbBuffer (PBYTE) - input
Bit-map data buffer.

Address in application storage from which the bit-map data is to be copied. This field must point to a doubleword aligned address to assure correct creation of the bit map by the device in device specific format.

pbmiInfoTable (PBITMAPINFO2) - input
Bit-map information table.

lScansSet (LONG) - returns
Number of scan lines actually set.

>=0
Number of scan lines actually set
GPI_ALTERROR
Error.

GpiSetBitmapBits - Remarks

The presentation space must be currently associated with a memory device context that has a bit map currently selected.

Note: This function does not set bits directly to any other kind of device.

If the format of the supplied bit map does not match that of the device, it is converted, using the supplied bit-map information table. The

standard bit-map formats are supported, plus any known to be supported by the device; see GpiQueryDeviceBitmapFormats.

GpiSetBitmapBits - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_INFO_TABLE (0x208F)
An invalid bit-map info table was specified with a bit-map operation.

PMERR_NO_BITMAP_SELECTED (0x20E4)
An attempt has been made to operate on a memory device context that has no bit map selected.

PMERR_INV_SCAN_START (0x20C4)
An invalid scanstart parameter was specified with a bit-map function.

PMERR_INCORRECT_DC_TYPE (0x203C)
An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

GpiSetBitmapBits - Related Functions

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapDimension
- GpiSetBitmapId
- GpiWCBitBlt

GpiSetBitmapBits - Example Code

This example uses the GpiSetBitmapBits function to copy image data to a bit map in a presentation space associated with a memory device context.

```
#define INCL_GPIPRIMITIVES
#define INCL_GPIBITMAPS
#define INCL_DOSMEMMGR
```

```
#define INCL_WINDIALOGS
#include <OS2.H>

 HPS     hps;                        /* Presentation space handle */
 BITMAPINFOHEADER2 bmp = { 12, 32, 16, 1, 1 };
 SEL sel;
 PBITMAPINFOHEADER2 pbmi;
 BYTE pbBuffer[16];     /* buffer for image data */
 ULONG cbBitmapInfo;
 HBITMAP hbm, hbmOld;
 BOOL f;
 /* Allocate space for the bit-map information table. */

cbBitmapInfo = sizeof(BITMAPINFO) + sizeof(RGB2);
f = (BOOL)DosAllocMem((PPVOID)pbmi,
                      (ULONG)cbBitmapInfo,
                      PAG_READ |
                      PAG_WRITE |
                      PAG_COMMIT));
if (f) {
        WinMessageBox(HWND_DESKTOP, HWND_DESKTOP,
                      "Sorry, Not enough memory",
                      NULL,
                      0,
                      MB_OK);
        return;
       }

 /* Initialize the bit-map information table. */

 pbmi->cbFix = 12;
 pbmi->cx = 32;
 pbmi ->cy = 16;
 pbmi->cPlanes = 1;
 pbmi ->cBitCount = 1;

 /* Create the bit map, set to the device,
    and copy the image data. */

 hbm = GpiCreateBitmap(hps,
                       pbmi,
                       0L,
                       NULL,
                       NULL);

 hbmOld = GpiSetBitmap(hps, hbm);     /* Save the old bitmap */

 GpiSetBitmapBits(hps,
                  0L,
                  (LONG)bmp.cy,
                  pbBuffer,
                  pbmi);
```

GpiSetBitmapBits - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetBitmapDimension

GpiSetBitmapDimension - Syntax

This function associates a width and height with a bit map, in units of 0.1 millimeter.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HBITMAP    hbm;                  /*  Bit-map handle. */
PSIZEL     psizlBitmapDimension; /*  Width and height of bit map. */
BOOL       rc;                   /*  Success indicator. */

rc = GpiSetBitmapDimension(hbm, psizlBitmapDimension);
```

GpiSetBitmapDimension Parameter - hbm

hbm (HBITMAP) - input
Bit-map handle.

GpiSetBitmapDimension Parameter - psizlBitmapDimension

psizlBitmapDimension (PSIZEL) - input
Width and height of bit map.

The width and height, respectively, of the bit map in units of 0.1 millimeter.

GpiSetBitmapDimension Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetBitmapDimension - Parameters

hbm (HBITMAP) - input
Bit-map handle.

psizlBitmapDimension (PSIZEL) - input
Width and height of bit map.

The width and height, respectively, of the bit map in units of 0.1 millimeter.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetBitmapDimension - Remarks

The values set are not used internally by the system, but are retained with the bit map and can be retrieved with GpiQueryBitmapDimension.

GpiSetBitmapDimension - Errors

Possible returns from WinGetLastError

PMERR_INV_HBITMAP (0x207B)
An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY (0x2032)
An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiSetBitmapDimension - Related Functions

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapId

- GpiWCBitBlt

GpiSetBitmapDimension - Example Code

This example uses the GpiSetBitmap and GpiSetBitmapDimension function to set a newly created bit map in the device context for the associated presentation space. Once set, the example initializes the bit-map image by drawing in the presentation space.

```
#define INCL_GPIBITMAPS
#define INCL_GPIPRIMITIVES
#include <OS2.H>

 HPS     hps;                       /* Presentation space handle */
 BOOL    fSuccess;                  /* Success indicator         */
 BITMAPINFOHEADER2 bmp = {12, 64, 64, 1, 1};/* 64x64 mono bit map */
 HBITMAP hbm, hbmOld;
 POINTL ptlStart = { 0, 0 };
 POINTL aptlTriangle[3] = { 32, 32, 63, 63, 0, 0 };
 POINTL ptl = { 63, 63 };
SIZEL sizlBitmapDimension = {100, 100};
                            /* The width and height, */
                            /* respectively, of the bit */
                            /* map in units of 0.1      */
                            /* millimeter.              */
 hbm = GpiCreateBitmap(hps,
                       &bmp,
                       0L,
                       NULL,
                       NULL);

 /* Set the bit map and draw in it. */

 fSuccess = GpiSetBitmapDimension(hbm,
                                  &sizlBitmapDimension);
 hbmOld = GpiSetBitmap(hps, hbm);        /* sets bit map
                                            in device context */
 GpiMove(hps, &ptlStart);
 GpiBox(hps, DRO_FILL, &ptl, 0L, 0L); /* fills in the bit map */
 GpiPolyLine(hps,                     /* draws a triangle        */
             1L,
             aptlTriangle);
 GpiSetBitmap(hps, hbmOld);           /* restores the old bit map*/
```

GpiSetBitmapDimension - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetBitmapId

GpiSetBitmapId - Syntax

This function tags a bit map with a local identifier, so that it can be used as a pattern set, containing a single member.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;    /*  Presentation-space handle. */
HBITMAP    hbm;    /*  Bit-map handle. */
LONG       lLcid;  /*  Local identifier with which the bit map is to be tagged. */
BOOL       rc;     /*  Success indicator. */

rc = GpiSetBitmapId(hps, hbm, lLcid);
```

GpiSetBitmapId Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetBitmapId Parameter - hbm

hbm (HBITMAP) - input
Bit-map handle.

The bit map must not be currently selected into a device context (see GpiSetBitmap).

GpiSetBitmapId Parameter - lLcid

lLcid (LONG) - input
Local identifier with which the bit map is to be tagged.

Valid values are in the range 1 through 254.

It is an error if the local identifier is already used to refer to a font or bit map.

GpiSetBitmapId Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetBitmapId - Parameters

hps (HPS) - input
Presentation-space handle.

hbm (HBITMAP) - input
Bit-map handle.

The bit map must not be currently selected into a device context (see GpiSetBitmap).

lLcid (LONG) - input
Local identifier with which the bit map is to be tagged.

Valid values are in the range 1 through 254.

It is an error if the local identifier is already used to refer to a font or bit map.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetBitmapId - Remarks

To use the bit map for area shading (or as the pattern in a GpiBitBlt or GpiWCBitBlt operation), a GpiSetPatternSet must be issued with the specified local identifier.

Any bit map of a format supported by the device can be specified. However, it may be simplified before use (see GpiSetPatternSet).

GpiDeleteSetId can subsequently be used to release the tag.

GpiSetBitmapId - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HBITMAP (0x207B)
An invalid bit-map handle was specified.

PMERR_INV_SETID (0x20CA)
An invalid setid parameter was specified.

PMERR_SETID_IN_USE (0x2102)
An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set.

PMERR_BITMAP_IN_USE (0x2008)
An attempt was made either to set a bit map into a device context using GpiSetBitmap while it was already selected into an existing device context, or to tag a bit map with a local pattern set identifier (setid) using GpiSetBitmapId while it was already tagged with an existing setid.

PMERR_HBITMAP_BUSY (0x2032)
An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiSetBitmapId - Related Functions

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDeleteSetId
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetPatternSet
- GpiWCBitBlt

GpiSetBitmapId - Example Code

This function tags a bit map with a local identifier, so that it can be used as a pattern set, containing a single member.

```
#define INCL_GPIBITMAPS
#include <OS2.H>

 HPS     hps;    /* Presentation space handle */
 HBITMAP hbm;    /* bit-map handle. */
 LONG lid = 23;  /* local identifier. */

GpiSetBitmapId(hps,
               hbm,
               lid);
```

GpiSetBitmapId - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetCharAngle

GpiSetCharAngle - Syntax

This function specifies the angle of the baseline for the characters in a string, as a relative vector.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS           hps;          /*  Presentation-space handle. */
PGRADIENTL    pgradlAngle;  /*  Baseline angle. */
BOOL          rc;           /*  Success indicator. */

rc = GpiSetCharAngle(hps, pgradlAngle);
```

GpiSetCharAngle Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCharAngle Parameter - pgradlAngle

pgradlAngle (PGRADIENTL) - input
Baseline angle.

The baseline angle is defined in terms of the relative coordinates of the point *pgradlAngle* (*x*, *y*).

If both *x* and *y* are **0**, the character angle is reset to the default value. This default value is (1,0), unless changed with GpiSetDefAttrs.

GpiSetCharAngle Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharAngle - Parameters

hps (HPS) - input
Presentation-space handle.

pgradlAngle (PGRADIENTL) - input
Baseline angle.

The baseline angle is defined in terms of the relative coordinates of the point *pgradlAngle* (*x*, *y*).

If both *x* and *y* are **0**, the character angle is reset to the default value. This default value is (1,0), unless changed with GpiSetDefAttrs.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharAngle - Remarks

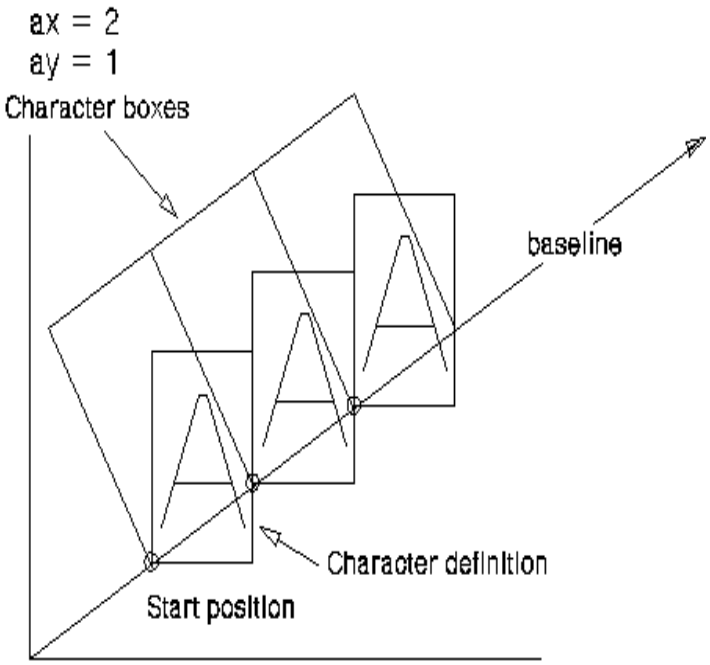
The coordinates of the point *pgradlAngle* specify integer values for the coordinates of the end of a line starting at the origin (0,0); the base line for subsequent character strings is parallel to this line.

The effect of the baseline angle attribute depends on the value of the character mode attribute (see GpiSetCharMode), and whether the current font is an outline or a raster font, as described below.

When the character mode is set to CM_MODE1, and the current font is a raster font, the character angle can be ignored.

When the character mode is set to CM_MODE2, and the current font is a raster font, the angle is used to determine the position of each character, but the orientations of characters within the character box may not be affected by changes in character angle. If this is so, the characters are positioned so that the lower left-hand corners of the character definitions are placed at the lower left-hand corners of the character boxes after all transforms have been applied. This is illustrated below.

For illustrative purposes, the figure shows all character reference points at their bottom left-hand corner.



When the character mode is set to CM_MODE3, or when the current font is an outline font, the angle is observed accurately, and the character boxes are rotated to be normal (perpendicular) to the character baseline. If the world coordinate system is such that one x-axis unit is not physically equal to one y-axis unit, a rotated character string appears to be sheared.

This function must not be issued in an area bracket.

The attribute mode determines whether the current value of the baseline angle attribute is preserved.

GpiSetCharAngle - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiSetCharAngle - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiPop
- GpiQueryCharAngle
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiSetAttrMode
- GpiSetAttrs

- GpiSetBackColor
- GpiSetBackMix
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix

GpiSetCharAngle - Graphic Elements and Orders

Element Type: **OCODE_GSCA**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Character Angle

Element Type: **OCODE_GPSCA**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Set Character Angle/Push and Set Character Angle

GpiSetCharAngle - Example Code

This function resets the angle of the baseline for the characters in a string, as a relative vector.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;
GRADIENTL gradlAngle = {0L, 0L};
GpiSetCharAngle(hps,
                &gradlAngle);
```

GpiSetCharAngle - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetCharBox

GpiSetCharBox - Syntax

This function sets the current character-box attribute to the specified value.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
PSIZEF    psizfxBox;  /*  Character-box size in world coordinates. */
BOOL      rc;         /*  Success indicator. */

rc = GpiSetCharBox(hps, psizfxBox);
```

GpiSetCharBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCharBox Parameter - psizfxBox

psizfxBox (PSIZEF) - input
Character-box size in world coordinates.

The width determines the spacing of consecutive characters along the baseline.

Both width and height can be positive, negative, or zero.

When either parameter is negative, the spacing occurs in the opposite direction to normal and each character is drawn reflected in character-mode 3. Thus, for example, a negative height in the standard direction in mode 3 means that the characters are drawn upside down, and the string drawn below the baseline (assuming no other transformations cause inversion).

A zero character width or height is also valid; here, the string of characters becomes a line. If both are zero, the string is drawn as a single point.

GpiSetCharBox Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharBox - Parameters

hps (HPS) - input
Presentation-space handle.

psizfxBox (PSIZEF) - input
Character-box size in world coordinates.

The width determines the spacing of consecutive characters along the baseline.

Both width and height can be positive, negative, or zero.

When either parameter is negative, the spacing occurs in the opposite direction to normal and each character is drawn reflected in character-mode 3. Thus, for example, a negative height in the standard direction in mode 3 means that the characters are drawn upside down, and the string drawn below the baseline (assuming no other transformations cause inversion).

A zero character width or height is also valid; here, the string of characters becomes a line. If both are zero, the string is drawn as a single point.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharBox - Remarks

The parameter *psizfxBox* specifies values for the width and height of a character box in world coordinates.

Whether these values are used when character strings are drawn depends on the type of font being used (raster or outline), and on the value of the character mode attribute (see the GpiSetCharMode function).

For raster fonts, where the character box is used only for positioning in character mode CM_MODE2, the character box width corresponds to the *lEmInc* font metric (see FONTMETRICS). For proportionally-spaced raster fonts, the effective spacing for a given character is the character box width, scaled by the ratio of that character's width to *lEmInc*.

For outline fonts, characters are defined in font definition space. The *sXDeviceRes* and *sYDeviceRes* fields (see FONTMETRICS) define a rectangle in font definition space that is mapped to the character box rectangle (modified by the character angle and shear attributes) in world coordinates. *sYDeviceRes* corresponds to the font point size in font definition space, and therefore the character box height corresponds to the font point size in world coordinates. This is typically less than the *lMaxBaselineExt*.

The effective width of each character from an outline font is the character box width, scaled by the ratio of the width of the character to *sXDeviceRes*. The *lAveCharWidth* (for a proportionally-spaced font) will therefore typically be smaller than the character box width. Indeed, because of differences in font design, *lAveCharWidth* and *lMaxBaselineExt* vary between different fonts, even when the character box dimensions are the same.

lEmInc and *lEmHeight* are always equal to the character box width and height, respectively.

To cause characters of a given point-size to be generated using an outline font, establish a world coordinate space with specific metrics (for

example, specify PU_TWIPS on GpiCreatePS), and set the character box height to the required point size. Because *sXDeviceRes* and *sYDeviceRes* are normally equal, the character box width should also be set equal to the height, if characters are required with the same aspect ratio as defined in the font (assuming that world coordinate space is *isotropic*).

The initial default value of the character box is the device-coordinates value returned by DevQueryCaps (CAPS_GRAPHICS_CHAR_WIDTH and CAPS_GRAPHICS_CHAR_HEIGHT) for the currently associated device, converted to page coordinates.

Note: In general the initial default value is rectangular, and to avoid character distortion, the character box should normally be set explicitly to be square if an outline font might be used (assuming that world coordinate space is isotropic).

The default value can be changed with GpiSetDefAttrs.

GpiSetCharBox must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character-box size attribute is preserved.

GpiSetCharBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiSetCharBox - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetCharAngle
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetMix
- GpiSetDefAttrs
- GpiQueryCharBox
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

GpiSetCharBox - Graphic Elements and Orders

Element Type: **OCODE_GSCC**

This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE. Note that GpiCreateLogFont can also generate this element type.

Order: Set Character Cell

Element Type: **OCODE_GPSCC**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Set Character Cell/Push and Set Character Cell

GpiSetCharBox - Example Code

This function sets the current character-box attribute to the specified value.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>


 HPS     hps;    /* Presentation space handle */
SIZEF sizfCharBox; /* Character-box size in  */
                   /* world coordinates.     */

  sizfCharBox.cx = MAKEFIXED(8,0);     /* values are shifted to the */
                                       /* left 16 bits              */
  sizfCharBox.cy = MAKEFIXED(20,0);    /* to make them fixed.       */
  GpiSetCharBox( hps,
                 &sizfCharBox );
```

GpiSetCharBox - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetCharBreakExtra

GpiSetCharBreakExtra - Syntax

This function specifies an extra increment to be used for spacing break characters in a string.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;         /*  Presentation-space handle. */
FIXED    BreakExtra;  /*  Character-break-extra value. */
BOOL     rc;          /*  Success indicator. */

rc = GpiSetCharBreakExtra(hps, BreakExtra);
```

GpiSetCharBreakExtra Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCharBreakExtra Parameter - BreakExtra

BreakExtra (FIXED) - input
Character-break-extra value.

The value can be negative, 0, or positive:

- A negative value reduces the effective width of break characters.
- A value of 0 results in normal spacing.
- A positive value increases the effective width of break characters.

The value is in world coordinates.

GpiSetCharBreakExtra Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharBreakExtra - Parameters

hps (HPS) - input
Presentation-space handle.

BreakExtra (FIXED) - input
Character-break-extra value.

The value can be negative, 0, or positive:

- A negative value reduces the effective width of break characters.
- A value of 0 results in normal spacing.
- A positive value increases the effective width of break characters.

The value is in world coordinates.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharBreakExtra - Remarks

The character-break-extra attribute provides a spacing value that increases or decreases the spacing for break characters in a string. The break character is defined by the font, and can be found by calling GpiQueryFonts (*sBreakChar* field in the FONTMETRICS structure).

The break-extra spacing is additional to the spacing generated for other reasons, for example:

- The spacing determined by the font, including proportional spacing and kerning, if applicable
- The vector of increment values, see GpiCharStringPos, GpiCharStringPosAt, GpiQueryCharStringPos and GpiQueryCharStringPosAt
- Extra spacing, see GpiSetCharExtra.

Break-extra spacing applies to character strings either within or outside a path definition (see GpiBeginPath). The effect of the character-break-extra attribute applies whatever the value of the character-mode attribute (see GpiSetCharMode), and for both outline and *raster* fonts.

This function must not be issued in an area bracket.

The initial default value of the character-break-extra attribute is 0, which gives normal spacing. This default value can be changed with GpiSetDefAttrs.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character-break-extra attribute is preserved.

GpiSetCharBreakExtra - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiSetCharBreakExtra - Graphic Elements and Orders

Element Type: **OCODE_GSCBE**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Character Break Extra

Element Type: **OCODE_GPSCBE**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Character Break Extra

GpiSetCharBreakExtra - Example Code

This function specifies an extra increment to be used for spacing break characters in a string.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>


 HPS     hps;    /* Presentation space handle     */
 FIXED  fxBreak; /* Character-break-extra  value. */
                 /* world coordinates.            */

  fxBreak = 8L<<16;            /* values are shifted to the */
                               /* left 16 bits              */
                               /* to make them fixed.       */

  GpiSetCharBreakExtra(hps,
                       fxBreak);
```

GpiSetCharBreakExtra - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Graphic Elements and Orders
Glossary

GpiSetCharDirection

GpiSetCharDirection - Syntax

This function determines the direction in which the characters in a string are drawn relative to the baseline.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;         /*  Presentation-space handle. */
LONG    lDirection;  /*  Character direction. */
BOOL    rc;          /*  Success indicator. */

rc = GpiSetCharDirection(hps, lDirection);
```

GpiSetCharDirection Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCharDirection Parameter - lDirection

lDirection (LONG) - input
Character direction.

CHDIRN_DEFAULT
The default; the same as CHDIRN_LEFTRIGHT (unless changed with GpiSetDefAttrs)

CHDIRN_LEFTRIGHT
Character boxes are arranged parallel to, and in the same direction as, the baseline. This is the usual convention for Roman text.

CHDIRN_TOPBOTTOM
Character boxes are arranged in columns directed 90 degrees *clockwise* from the baseline. This is the usual convention for Chinese characters. This option can be used for drawing Roman text vertically (a y-axis title on a graph, for example). The reference point within the character definition is at the center of the character, along the x-direction, in this case.

CHDIRN_RIGHTLEFT
Character boxes are arranged parallel to, but in the reverse of, the baseline direction. This is the usual convention for Arabic text.

CHDIRN_BOTTOMTOP
Character boxes are arranged in columns directed 90 degrees *counterclockwise* from the baseline. The reference point within the character definition is at the center of the character, along the x-direction, in this case.

GpiSetCharDirection Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE

Successful completion

FALSE
Error occurred.

GpiSetCharDirection - Parameters

hps (HPS) - input
Presentation-space handle.

lDirection (LONG) - input
Character direction.

CHDIRN_DEFAULT
The default; the same as CHDIRN_LEFTRIGHT (unless changed with GpiSetDefAttrs)

CHDIRN_LEFTRIGHT
Character boxes are arranged parallel to, and in the same direction as, the baseline. This is the usual convention for Roman text.

CHDIRN_TOPBOTTOM
Character boxes are arranged in columns directed 90 degrees *clockwise* from the baseline. This is the usual convention for Chinese characters. This option can be used for drawing Roman text vertically (a y-axis title on a graph, for example). The reference point within the character definition is at the center of the character, along the x-direction, in this case.

CHDIRN_RIGHTLEFT
Character boxes are arranged parallel to, but in the reverse of, the baseline direction. This is the usual convention for Arabic text.

CHDIRN_BOTTOMTOP
Character boxes are arranged in columns directed 90 degrees *counterclockwise* from the baseline. The reference point within the character definition is at the center of the character, along the x-direction, in this case.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiSetCharDirection - Remarks

This function must not be issued in an area bracket. The attribute mode determines whether the current value of the character direction attribute is preserved. The following diagram shows how the origin of characters changes when the direction changes:

```
  A   B   C                C   B   A
.           .            .           .
start       end          end         start

CHDIRN_LEFTRIGHT         CHDIRN_RIGHTLEFT


start                    end
  .                        .
  A                        C
```

```
  B                       B
  C                       A
  .                       .
end                     start
CHDIRN_TOPBOTTOM        CHDIRN_BOTTOMTOP
```

GpiSetCharDirection - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_DIRECTION_ATTR (0x204C)
: An invalid character direction attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetCharDirection - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiPop
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetDefAttrs
- GpiQueryCharDirection
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

GpiSetCharDirection - Graphic Elements and Orders

Element Type: **OCODE_GSCD**
: This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

 Order: Set Character Direction

Element Type: **OCODE_GPSCD**

This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Character Direction

GpiSetCharDirection - Example Code

This function determines the direction in which the characters in a string are drawn relative to the baseline. In this example, the direction is reset to the default, or left-to-right.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS     hps;    /* Presentation space handle     */

GpiSetCharDirection(hps, CHDIRN_DEFAULT);
```

GpiSetCharDirection - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetCharExtra

GpiSetCharExtra - Syntax

This function specifies an extra increment to be used for spacing characters in a string.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;    /*  Presentation-space handle. */
FIXED    Extra;  /*  Character-extra value. */
BOOL     rc;     /*  Success indicator. */

rc = GpiSetCharExtra(hps, Extra);
```

GpiSetCharExtra Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCharExtra Parameter - Extra

Extra (FIXED) - input
Character-extra value.

The value can be negative, 0, or positive:

- A negative value forces the characters closer together.
- A value of 0 results in normal spacing.
- A positive value forces the characters further apart.

The value is in world coordinates.

GpiSetCharExtra Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharExtra - Parameters

hps (HPS) - input
Presentation-space handle.

Extra (FIXED) - input
Character-extra value.

The value can be negative, 0, or positive:

- A negative value forces the characters closer together.
- A value of 0 results in normal spacing.
- A positive value forces the characters further apart.

The value is in world coordinates.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharExtra - Remarks

The character-extra attribute provides a spacing value that increases or decreases the spacing between characters in a string. It applies to all characters in a font, including the break character and is additional to the spacing generated for other reasons, for example:

- The spacing determined by the font, including proportional spacing and kerning, if applicable
- The vector of increment values, see GpiCharStringPos, GpiCharStringPosAt, GpiQueryCharStringPos and GpiQueryCharStringPosAt
- Break-extra spacing, see GpiSetCharBreakExtra.

Extra spacing applies to character strings either within or outside a path definition (see GpiBeginPath). The effect of the character-extra attribute applies whatever the value of the character-mode attribute (see GpiSetCharMode), and for both outline and *raster* fonts.

This function must not be issued in an area bracket.

The initial default value of the character-extra attribute is 0, which gives normal spacing. This default value can be changed with GpiSetDefAttrs.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character-extra attribute is preserved.

GpiSetCharExtra - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiSetCharExtra - Graphic Elements and Orders

Element Type: **OCODE_GSCE**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Character Extra

Element Type: **OCODE_GPSCE**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Character Extra

GpiSetCharExtra - Example Code

This function specifies an extra increment to be used for spacing characters in a string.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */
FIXED fxExtra;  /* extra character.

fxExtra = MAKEFIXED(0,0)      /* normal spacing. */
GpiSetCharExtra(hps,fxExtra);
```

GpiSetCharExtra - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Graphic Elements and Orders
Glossary

GpiSetCharMode

GpiSetCharMode - Syntax

This function controls the character mode used when drawing a character string.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lMode;  /*  Character mode. */
BOOL    rc;     /*  Success indicator. */

rc = GpiSetCharMode(hps, lMode);
```

GpiSetCharMode Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCharMode Parameter - lMode

lMode (LONG) - input
Character mode.

This parameter can have one of the following values:

CM_DEFAULT
The default; the same as CM_MODE1 (unless changed with GpiSetDefAttrs).

CM_MODE1
The font selected by means of GpiSetCharSet can be either a raster font or an outline font.

If it is a raster font, the position of characters after the first character is determined by the font metrics information, and by the character direction, character extra, and character break extra attributes. If it is an outline font, the behavior is as if the character mode is CM_MODE3.

CM_MODE2
The font selected by means of GpiSetCharSet can be either a raster font or an outline font.

If it is a raster font, the position of characters after the first character is determined by the font metrics information, and some character attributes, namely, GpiSetCharAngle, GpiSetCharBox, GpiSetCharDirection, GpiSetCharExtra, GpiSetCharBreakExtra, and GpiSetCharShear (the latter is relevant only for character directions of CHDIRN_TOPBOTTOM and CHDIRN_BOTTOMTOP). If it is an outline font, the behavior is as if the character mode is CM_MODE3.

CM_MODE3
All character attributes are used for positioning (together with the positioning information in the font), and for scaling, rotating, and shearing the characters.

The font selected by means of GpiSetCharSet must be an outline font; an error is raised if an attempt is made to draw a character string in CM_MODE3, and the selected font is a raster font.

GpiSetCharMode Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharMode - Parameters

hps (HPS) - input
Presentation-space handle.

lMode (LONG) - input
Character mode.

This parameter can have one of the following values:

CM_DEFAULT
The default; the same as CM_MODE1 (unless changed with GpiSetDefAttrs).

CM_MODE1
The font selected by means of GpiSetCharSet can be either a raster font or an outline font.

If it is a raster font, the position of characters after the first character is determined by the font metrics information, and by the character direction, character extra, and character break extra attributes. If it is an outline font, the behavior is as if the character mode is CM_MODE3.

CM_MODE2
The font selected by means of GpiSetCharSet can be either a raster font or an outline font.

If it is a raster font, the position of characters after the first character is determined by the font metrics information, and some character attributes, namely, GpiSetCharAngle, GpiSetCharBox, GpiSetCharDirection, GpiSetCharExtra, GpiSetCharBreakExtra, and GpiSetCharShear (the latter is relevant only for character directions of CHDIRN_TOPBOTTOM and CHDIRN_BOTTOMTOP). If it is an outline font, the behavior is as if the character mode is CM_MODE3.

CM_MODE3
All character attributes are used for positioning (together with the positioning information in the font), and for scaling, rotating, and shearing the characters.

The font selected by means of GpiSetCharSet must be an outline font; an error is raised if an attempt is made to draw a character string in CM_MODE3, and the selected font is a raster font.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiSetCharMode - Remarks

The value of the *lMode* parameter controls whether the character attributes affect the positioning of characters, as follows:

Use of Character Attributes in each Character Mode

Character Mode	Font Type	Character Attributes (Angle, Shear, and Box)
Mode 1	Raster	Ignored
	Outline	Used
Mode 2	Raster	Attribute information is used to position characters; characters are not sheared, rotated, or scaled.
	Outline	Used
Mode 3	Raster	An error is raised when an attempt is made to draw a character string.

```
Outline     Used
```

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character-mode attribute is preserved.

GpiSetCharMode - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_MODE_ATTR (0x204D)
: An invalid character mode attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetCharMode - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiPop
- GpiSetAttrs
- GpiSetAttrMode
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharSet
- GpiSetCharShear
- GpiSetDefAttrs
- GpiQueryCharMode
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

GpiSetCharMode - Graphic Elements and Orders

Element Type: **OCODE_GSCR**
: This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

 Order: Set Character Precision

Element Type: **OCODE_GPSCR**
: This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Character Precision

GpiSetCharMode - Example Code

In this example the GpiSetCharMode call is used to set the character mode to raster or outline when drawing a string.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */

GpiSetCharMode(hps,
               CM_MODE3); /* The font selected by    */
                          /* means of                */
                          /* GpiSetCharSet can be    */
                          /* either a raster font or */
                          /* an outline font.        */
```

GpiSetCharMode - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetCharSet

GpiSetCharSet - Syntax

This function sets the current value of the character-set attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    llcid;  /*  Character-set local identifier. */
BOOL    rc;     /*  Success indicator. */
```

```
rc = GpiSetCharSet(hps, llcid);
```

GpiSetCharSet Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCharSet Parameter - llcid

llcid (LONG) - input
Character-set local identifier.

LCID_DEFAULT
Default (can be set explicitly with GpiSetDefAttrs).
1-254
Identifies a logical font.

GpiSetCharSet Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCharSet - Parameters

hps (HPS) - input
Presentation-space handle.

llcid (LONG) - input
Character-set local identifier.

LCID_DEFAULT
Default (can be set explicitly with GpiSetDefAttrs).
1-254
Identifies a logical font.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetCharSet - Remarks

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character-set attribute is preserved.

GpiSetCharSet - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_SET_ATTR (0x204F)
: An invalid character setid attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_HUGE_FONTS_NOT_SUPPORTED (0x2035)
: An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

GpiSetCharSet - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiCreateLogFont
- GpiPop
- GpiQueryCharSet
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharShear
- GpiSetDefAttrs

GpiSetCharSet - Graphic Elements and Orders

Element Type: **OCODE_GSCS**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Character Set

Element Type: **OCODE_GPSCS**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Character Set

GpiSetCharSet - Example Code

This function sets the current value of the character-set attribute.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */
LONG llcid = 32L;

GpiSetCharSet(hps, llcid);
```

GpiSetCharSet - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetCharShear

GpiSetCharShear - Syntax

This function sets the character-shear attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlAngle;  /*  Character shear vector. */
BOOL       rc;         /*  Success indicator. */

rc = GpiSetCharShear(hps, pptlAngle);
```

GpiSetCharShear Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCharShear Parameter - pptlAngle

pptlAngle (PPOINTL) - input
Character shear vector.

With reference to the figure in the Notes, the shear angle is defined in terms of the relative coordinates of the point *pptlAngle* . (x, y).

If x is 0 and y is 1 (initial default), "upright" characters result. If x and y are both positive or both negative, the characters slope from bottom-left to top-right. If x and y are of opposite signs, the characters slope from top-left to bottom-right. No character inversion ever takes place as a result of a shear alone.

Usually, it is an error to specify 0 for y, because this implies an "infinite" shear. However, if both x and y are 0, the attribute is set to the default value. This can be changed from the initial default with GpiSetDefAttrs.

GpiSetCharShear Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetCharShear - Parameters

hps (HPS) - input

Presentation-space handle.

pptlAngle (PPOINTL) - input
Character shear vector.

With reference to the figure in the Notes, the shear angle is defined in terms of the relative coordinates of the point *pptlAngle* . (*x*, *y*).

If *x* is 0 and *y* is 1 (initial default), "upright" characters result. If *x* and *y* are both positive or both negative, the characters slope from bottom-left to top-right. If *x* and *y* are of opposite signs, the characters slope from top-left to bottom-right. No character inversion ever takes place as a result of a shear alone.

Usually, it is an error to specify 0 for *y*, because this implies an "infinite" shear. However, if both *x* and *y* are 0, the attribute is set to the default value. This can be changed from the initial default with GpiSetDefAttrs.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

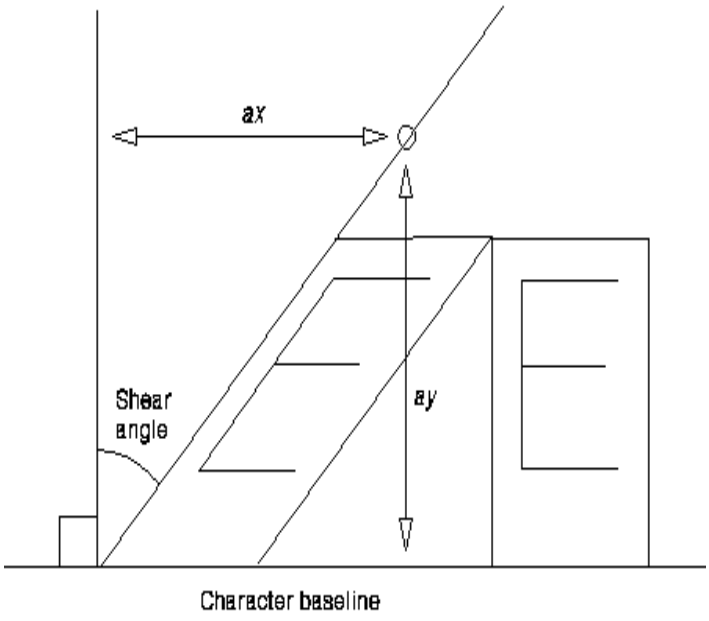
GpiSetCharShear - Remarks

The coordinates of the point *pptlAngle* (*x*, *y*), specify integer values that identify the end coordinates of a line originating at (0,0) (see figure below). The vertical strokes in subsequent character strings are drawn parallel to the defined line. The top of the character box remains parallel to the character baseline (which may itself be rotated).

Whether this attribute is used when character strings are drawn depends on the type of font being used (raster or outline), and on the value of the character mode attribute (see GpiSetCharMode). If it is used, then with character directions of CHDIRN_TOPBOTTOM and CHDIRN_BOTTOMTOP (see GpiSetCharDirection) the whole string is tilted by the shear angle, in addition to the individual characters being sheared if the current font is an outline font.

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the character shear attribute is preserved.



GpiSetCharShear - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_SHEAR_ATTR (0x2050)
An invalid character shear attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetCharShear - Related Functions

Related Functions

- GpiPop
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiQueryCharShear
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

GpiSetCharShear - Graphic Elements and Orders

Element Type: **OCODE_GSCH**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Character Shear

Element Type: **OCODE_GPSCH**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Character Shear

GpiSetCharShear - Example Code

This function sets the character-shear attribute.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */
POINTL ptlAngle = {50L, 70L}; /* character shear vector. */


GpiSetCharShear(hps,
                &ptlAngle); /*  the shear                  */
                            /*  angle is defined in terms  */
                            /*  of the relative            */
                            /*  coordinates of the point   */
                            /*  pptlAngle. This can be     */
                            /*  changed from the initial   */
                            /*  default with               */
                            /*  GpiSetDefAttrs.            */
```

GpiSetCharShear - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetClipPath

GpiSetClipPath - Syntax

This function selects a path as the current clip path.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lPath;     /*  Path control flag. */
LONG    lOptions;  /*  Options. */
BOOL    rc;        /*  Success indicator. */
```

```
rc = GpiSetClipPath(hps, lPath, lOptions);
```

GpiSetClipPath Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetClipPath Parameter - lPath

lPath (LONG) - input
Path control flag.

0	The current clip path stops being the current clip path; the current clip path is to be reset to an infinite one (that is, no clipping).
1	The path that has been defined is to be intersected with the current clip path.

GpiSetClipPath Parameter - lOptions

lOptions (LONG) - input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

How to construct the path interior (see also GpiBeginArea):

SCP_ALTERNATE	Constructs interior in alternate mode (the default).
SCP_WINDING	Constructs interior in winding mode. This value must be selected if the path has been modified using GpiModifyPath.

Relationship to current clip path:

SCP_AND	Intersects the specified path with the current clip path. This must be specified if *lPath* is 1.
SCP_RESET	Resets the clip path to an infinite clip path (the default). This must be specified or defaulted if *lPath* is 0.

Include or exclude path:

SCP_INCL	Includes this path in the clip path (the default).
SCP_EXCL	Excludes this path in the clip path.

GpiSetClipPath Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetClipPath - Parameters

hps (HPS) - input
Presentation-space handle.

lPath (LONG) - input
Path control flag.

0 The current clip path stops being the current clip path; the current clip path is to be reset to an infinite one (that is, no clipping).
1 The path that has been defined is to be intersected with the current clip path.

lOptions (LONG) - input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

How to construct the path interior (see also GpiBeginArea):

SCP_ALTERNATE — Constructs interior in alternate mode (the default).

SCP_WINDING — Constructs interior in winding mode. This value must be selected if the path has been modified using GpiModifyPath.

Relationship to current clip path:

SCP_AND — Intersects the specified path with the current clip path. This must be specified if *lPath* is 1.

SCP_RESET — Resets the clip path to an infinite clip path (the default). This must be specified or defaulted if *lPath* is 0.

Include or exclude path:

SCP_INCL — Includes this path in the clip path (the default).

SCP_EXCL — Excludes this path in the clip path.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetClipPath - Remarks

The clip path (bound in device coordinates when the path is defined) is used for all subsequent drawing.

Any open figures within the path are closed automatically.

The boundaries of the area defined by the path are considered to be part of the interior, so that a point on the boundary is not clipped.

The clip path is reset to no clipping (no path selected) at the start of a root segment (subject to the fast chaining segment attribute), or when a GpiResetPS function is issued.

After a path is selected as the clip path, it cannot be reused for any other purpose. When it is superseded as the clip path, it is discarded.

GpiSetClipPath - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID (0x20AE)
An invalid path identifier parameter was specified.

PMERR_INV_CLIP_PATH_OPTIONS (0x2051)
An invalid options parameter was specified with GpiSetClipPath.

PMERR_PATH_UNKNOWN (0x20EE)
An attempt was made to perform a path function on a path that did not exist.

GpiSetClipPath - Related Functions

Related Functions

- GpiBeginPath
- GpiEndPath
- GpiExcludeClipRectangle
- GpiFillPath
- GpiIntersectClipRectangle
- GpiModifyPath
- GpiOffsetClipRegion
- GpiOutlinePath
- GpiPathToRegion
- GpiStrokePath
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiSetClipPath
- GpiSetClipRegion

GpiSetClipPath - Graphic Elements and Orders

Element Type: **OCODE_GSCPTH**

Order: Set Clip Path

GpiSetClipPath - Example Code

This function selects a path as the current clip path.

```
#define INCL_GPIPATHS
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */


GpiSetClipPath(hps,
               0L,    /* The current clip path    */
                      /* stops being the          */
                      /* current clip path; the   */
                      /* current clip path is to  */
                      /* be reset to an infinite  */
                      /* one (that is, no         */
                      /* clipping)                */
               SCP_ALTERNATE);
                      /* Construct interior in    */
                      /* alternate mode.          */
```

GpiSetClipPath - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetClipRegion

GpiSetClipRegion - Syntax

This function defines the region to be used for clipping, when any drawing takes place through the specified presentation space.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;          /*  Presentation-space handle. */
HRGN     hrgn;         /*  Region handle. */
PHRGN    phrgnOld;     /*  Old region handle (if any). */
LONG     lComplexity;  /*  Complexity of clipping and error indicators. */

lComplexity = GpiSetClipRegion(hps, hrgn,
                phrgnOld);
```

GpiSetClipRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

The presentation space must be currently associated with a device context of the correct device class (defined when the region is first created).

GpiSetClipRegion Parameter - hrgn

hrgn (HRGN) - input
Region handle.

If *hrgn* is null, the clipping region is set to no clipping (its initial state).

GpiSetClipRegion Parameter - phrgnOld

phrgnOld (PHRGN) - output
Old region handle (if any).

HRGN_ERROR
Error
NULLHANDLE
Null handle (no region selected).
Otherwise
Old region handle.

GpiSetClipRegion Return Value - lComplexity

lComplexity (LONG) - returns
Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

The possible values for the return value are:

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiSetClipRegion - Parameters

hps (HPS) - input
Presentation-space handle.

The presentation space must be currently associated with a device context of the correct device class (defined when the region is first created).

hrgn (HRGN) - input
Region handle.

If *hrgn* is null, the clipping region is set to no clipping (its initial state).

phrgnOld (PHRGN) - output
Old region handle (if any).

HRGN_ERROR
: Error

NULLHANDLE
: Null handle (no region selected).

Otherwise
: Old region handle.

lComplexity (LONG) - returns
Complexity of clipping and error indicators.

The clipping complexity information includes the combined effects of:

- Clip path
- Viewing limits
- Graphics field
- Clip region
- Visible region (windowing considerations).

The possible values for the return value are:

RGN_NULL
: Null region

RGN_RECT
: Rectangular region

RGN_COMPLEX
: Complex region

RGN_ERROR
: Error.

GpiSetClipRegion - Remarks

While a region is in use as a clip region, the calls GpiOffsetClipRegion, GpiExcludeClipRectangle and GpiIntersectClipRectangle cause it to be changed. These changes persist after the region has been deselected. The clip region cannot, however, be used for any other region operations, nor can it be selected into any other presentation space as a clipping region, until it is deselected.

The coordinates of the region are taken to be device coordinates within the device context.

The previous clip region, if any, is converted to a region, and a handle to it is returned. This can be used in a subsequent GpiSetClipRegion to reinstate the same clipping as before. If no clip region exists, a null handle is returned. It is the responsibility of the application to free the previous clip region (if any) with GpiDestroyRegion, even if this region was not originally created explicitly by the application.

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetClipRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiSetClipRegion - Related Functions

Related Functions

- GpiCreateRegion
- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiPtVisible
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiRectVisible

GpiSetClipRegion - Topics

Select an item:
Syntax

Parameters
Returns
Errors
Remarks
Related Functions
Glossary

GpiSetColor

GpiSetColor - Syntax

This function sets the current value of the color attribute for each of the individual primitive types.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lColor;  /*  Color. */
BOOL    rc;      /*  Success indicator. */

rc = GpiSetColor(hps, lColor);
```

GpiSetColor Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetColor Parameter - lColor

lColor (LONG) - input
Color.

It can be either an index value, if the color table is in index mode, or an RGB value, if the color table is in RGB mode. For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

The possible values include:

CLR_FALSE
All color planes are 0s.

CLR_TRUE
All color planes are 1s.

CLR_DEFAULT
: Set to default value. This is a device-dependent color which, for the default color table, provides a contrasting color to CLR_BACKGROUND For a display, it is the default window text color (SYSCR_WINDOWTEXT: see WinSetSysColors in the *Presentation Manager Programming Reference*). For a printer, it is a color that contrasts with the paper color. The default can be changed by setting new system colors from the control panel for the display, or by selecting a paper color for a printer, if allowed by the device driver. It can also be set explicitly, using GpiSetDefAttrs.

CLR_WHITE
: White (default color table, or index=RGB loaded color table). For a loaded, realized, color table it is the nearest available color to white.

CLR_BLACK
: Black (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to black.

CLR_BACKGROUND
: Reset color, used by GpiErase. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see WinSetSysColors in the *Presentation Manager Programming Reference*) for the default color table. For a printer, it is the paper color. For a loaded palette, which overrides a loaded color table, it is the first entry in the palette. For an RGB color table, it is color 000000 (black).

CLR_BLUE
: Blue (default color table).

CLR_RED
: Red (default color table).

CLR_PINK
: Pink (default color table).

CLR_GREEN
: Green (default color table).

CLR_CYAN
: Cyan (default color table).

CLR_YELLOW
: Yellow (default color table).

CLR_NEUTRAL
: Neutral (default color table). A device-dependent color, that for the default color table provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT: see WinSetSysColors in the *Presentation Manager Programming Reference*). For a printer, it is a color that contrasts with the paper color. For a loaded color table, it is color index 7; in RGB mode it is interpreted as color 000007.

CLR_DARKGRAY
: Dark gray (default color table).

CLR_DARKBLUE
: Dark blue (default color table).

CLR_DARKRED
: Dark red (default color table).

CLR_DARKPINK
: Dark pink (default color table).

CLR_DARKGREEN
: Dark green (default color table).

CLR_DARKCYAN
: Dark cyan (default color table).

CLR_BROWN
: Brown (default color table).

CLR_PALEGRAY
: Pale gray (default color table).

GpiSetColor Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiSetColor - Parameters

hps (HPS) - input
Presentation-space handle.

lColor (LONG) - input
Color.

It can be either an index value, if the color table is in index mode, or an RGB value, if the color table is in RGB mode. For a loadable color table, values 0 through n correspond to the color index (or RGB) values.

The possible values include:

CLR_FALSE
All color planes are 0s.

CLR_TRUE
All color planes are 1s.

CLR_DEFAULT
Set to default value. This is a device-dependent color which, for the default color table, provides a contrasting color to CLR_BACKGROUND For a display, it is the default window text color (SYSCR_WINDOWTEXT: see WinSetSysColors in the *Presentation Manager Programming Reference*). For a printer, it is a color that contrasts with the paper color. The default can be changed by setting new system colors from the control panel for the display, or by selecting a paper color for a printer, if allowed by the device driver. It can also be set explicitly, using GpiSetDefAttrs.

CLR_WHITE
White (default color table, or index=RGB loaded color table). For a loaded, realized, color table it is the nearest available color to white.

CLR_BLACK
Black (default color table, or index=RGB loaded color table). For a loaded, realized, color table, it is the nearest available color to black.

CLR_BACKGROUND
Reset color, used by GpiErase. This is the natural background color for the device. For a display, it is the default window color (SYSCLR_WINDOW: see WinSetSysColors in the *Presentation Manager Programming Reference*) for the default color table. For a printer, it is the paper color. For a loaded palette, which overrides a loaded color table, it is the first entry in the palette. For an RGB color table, it is color 000000 (black).

CLR_BLUE
Blue (default color table).

CLR_RED
Red (default color table).

CLR_PINK
Pink (default color table).

CLR_GREEN

Green (default color table).

CLR_CYAN
: Cyan (default color table).

CLR_YELLOW
: Yellow (default color table).

CLR_NEUTRAL
: Neutral (default color table). A device-dependent color, that for the default color table provides a contrasting color to CLR_BACKGROUND. For a display, it is the default window text color (SYSCLR_WINDOWTEXT: see WinSetSysColors in the *Presentation Manager Programming Reference*). For a printer, it is a color that contrasts with the paper color. For a loaded color table, it is color index 7; in RGB mode it is interpreted as color 000007.

CLR_DARKGRAY
: Dark gray (default color table).

CLR_DARKBLUE
: Dark blue (default color table).

CLR_DARKRED
: Dark red (default color table).

CLR_DARKPINK
: Dark pink (default color table).

CLR_DARKGREEN
: Dark green (default color table).

CLR_DARKCYAN
: Dark cyan (default color table).

CLR_BROWN
: Brown (default color table).

CLR_PALEGRAY
: Pale gray (default color table).

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetColor - Remarks

The current values for each primitive type are updated. The attribute mode (see GpiSetAttrMode) determines whether the current values of the individual color attributes are preserved. If so, they are restored by a single GpiPop function.

An attempt to set a negative color value, other than one for which a constant is defined, causes the error PMERR_INV_COLOR_ATTR to be logged. Other color values are allowed, although an error is generated when the color value is needed for drawing if it is not valid for the color table in use at that time (see GpiCreateLogColorTable).

For details of how colors are handled on monochrome devices, see GpiCreateLogColorTable.

GpiSetColor - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COLOR_ATTR (0x2053)
An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetColor - Related Functions

Related Functions

- GpiQueryColor
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetDefAttrs
- GpiSetMix

GpiSetColor - Graphic Elements and Orders

Element Type: **OCODE_GSICOL**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Indexed Color

Element Type: **OCODE_GPSICOL**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Indexed Color

GpiSetColor - Example Code

This example draws a blue line.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */
POINTL ptl1, ptl2;

      GpiSetColor(hps, CLR_BLUE);
      GpiMove( hps, &ptl1 );               /* Move to start point           */
      GpiLine( hps, &ptl2 );               /* Draw new line                 */
```

GpiSetColor - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetCp

GpiSetCp - Syntax

This function sets the default graphics code page.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;         /*  Presentation-space handle. */
ULONG    ulCodePage;  /*  Code-page identifier. */
BOOL     rc;          /*  Success indicator. */

rc = GpiSetCp(hps, ulCodePage);
```

GpiSetCp Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCp Parameter - ulCodePage

ulCodePage (ULONG) - input
Code-page identifier.

GpiSetCp Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCp - Parameters

hps (HPS) - input
Presentation-space handle.

ulCodePage (ULONG) - input
Code-page identifier.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCp - Remarks

The default graphics code page is used for the default font (unless it is overridden by GpiCreateLogFont). It is also used for other fonts for which the *usCodePage* field in the FATTRS structure for GpiCreateLogFont is specified as 0. This includes existing fonts that are defined in this way.

Any code page that is defined in the CONFIG.SYS file, or is a supported EBCDIC code page, can be selected. The following is the list of valid code pages:

Country	Code Page
Canadian-French	863
Desktop Publishing	1004
Iceland	861
Latin 1 Multilingual	850
Latin 2 Multilingual	852
Nordic	865
Portuguese	860
Turkey	857
U.S. (IBM PC)	437

Code page 1004 is compatible with Microsoft** Windows**.

The following EBCDIC code pages, based on character set 697, are also available for output:

Country	Code Page
Austrian/German	273
Belgian	500

Brazil	037
Czechoslovakia	870
Danish/Norwegian	277
Finnish/Swedish	278
French	297
Hungary	870
Iceland	871
International	500
Italian	280
Poland	870
Portuguese	037
Spanish	284
Turkey	1026
U.K.-English	285
U.S.-English	037
Yugoslavia	870

Code pages 274 (Belgian) and 282 (Portuguese) can be used to provide access to old data.

When a GPI presentation space is first created, it uses code page 850 if available, otherwise it uses the current code page.

If this function occurs within a path definition when the drawing mode (see GpiSetDrawingMode) is **retain** or **draw-and-retain**, its effect is not stored with the definition.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetCp - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CODEPAGE (0x2052)
An invalid code-page parameter was specified with GpiSetCp.

GpiSetCp - Related Functions

Related Functions The DOS calls DosGetCp, DosSetCp, and DosSetProcCp are related to GpiSetCP, but they are not a part of the Presentation Manager, for more information on the mentioned DOS calls refer to the *Control Program Reference* .

- GpiCreateLogFont
- GpiQueryCurrentPosition

GpiSetCp - Example Code

This example sets the code page to 850.

```
#define INCL_GPILCIDS
```

```
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */
ULONG  ulCodePage = 850;

GpiSetCp(hps, ulCodePage);
```

GpiSetCp - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetCurrentPosition

GpiSetCurrentPosition - Syntax

This function sets the current position to the specified point.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlPoint;  /*  New value of current position. */
BOOL       rc;         /*  Success indicator. */

rc = GpiSetCurrentPosition(hps, pptlPoint);
```

GpiSetCurrentPosition Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetCurrentPosition Parameter - pptlPoint

pptlPoint (PPOINTL) - input
New value of current position.

GpiSetCurrentPosition Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCurrentPosition - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - input
New value of current position.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetCurrentPosition - Remarks

This function also has the effect of resetting the position within a line-type sequence, and, if within an area, of starting a new closed figure and causing any previous one to be closed if necessary.

This function is equivalent to the GpiMove function except that, if the current attribute mode is AM_PRESERVE (see GpiSetAttrMode), the current position is saved before being set to the new value, so that it can be restored using the GpiPop function.

GpiSetCurrentPosition - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetCurrentPosition - Related Functions

Related Functions

- GpiMove
- GpiQueryCurrentPosition
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs

GpiSetCurrentPosition - Graphic Elements and Orders

Element Type: **OCODE_GSCP**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Current Position

Element Type: **OCODE_GPSCP**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Current Position

GpiSetCurrentPosition - Example Code

The position of the top-left corner of the window rectangle is recorded and selected as the current position before the image is drawn.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
HPS     hps;    /* Presentation space handle     */
HWND hwndClient; /* client window handle. */
RECTL rcl;
POINTL vptlSave;


WinQueryWindowRect(hwndClient, &rcl);
vptlSave.x = rcl.xLeft;
vptlSave.y = rcl.yTop;
GpiSetCurrentPosition(hps, &vptlSave);
```

GpiSetCurrentPosition - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetDefArcParams

GpiSetDefArcParams - Syntax

This function specifies the default values of the arc parameters (see GpiSetArcParams).

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS           hps;             /*  Presentation-space handle. */
PARCPARAMS    parcpArcParams;  /*  Default arc parameters. */
BOOL          rc;              /*  Success indicator. */

rc = GpiSetDefArcParams(hps, parcpArcParams);
```

GpiSetDefArcParams Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetDefArcParams Parameter - parcpArcParams

parcpArcParams (PARCPARAMS) - input
Default arc parameters.

This structure has four elements p, q, r, and s.

GpiSetDefArcParams Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDefArcParams - Parameters

hps (HPS) - input
Presentation-space handle.

parcpArcParams (PARCPARAMS) - input
Default arc parameters.

This structure has four elements p, q, r, and s.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDefArcParams - Remarks

The arc parameters are reset to their default values at the following times:

- When the presentation space is associated with a device context (see GpiAssociate).
- When GpiResetPS is issued.
- When drawing of a chained segment begins or ends (see GpiOpenSegment and GpiCloseSegment for more details).

The initial default values of the arc parameters, when the presentation space is first created, are :

```
p = 1     r = 0
s = 0     q = 1
```

The default values can be changed by GpiSetDefArcParams. Changing the default values has an immediate effect on the current arc parameters, if these are currently set to the default value.

See GpiSetArcParams for a description of the arc parameters.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetDefArcParams - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
: An invalid coordinate value was specified.

GpiSetDefArcParams - Related Functions

Related Functions

- GpiFullArc
- GpiPartialArc
- GpiPointArc
- GpiQueryDefArcParams
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs

GpiSetDefArcParams - Example Code

This function specifies the default values of the arc parameters (see GpiSetArcParams).

```
#define INCL_GPIDEFAULTS
#define INCL_GPIPRIMITIVES
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */

ARCPARAMS ArcParams = {10L, /* p */
                       20L, /* q */
                       10L, /* r */
                       30L}; /* l */
                     /* This structure has four   */
                     /* elements p, q, r, and s.  */


GpiSetDefArcParams(hps, &ArcParams);
```

GpiSetDefArcParams - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetDefAttrs

GpiSetDefAttrs - Syntax

This function sets the default values of attributes for the specified primitive type.

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;         /*  Presentation-space handle. */
LONG       lPrimType;   /*  Primitive type. */
ULONG      flAttrMask;  /*  Attributes mask. */
PBUNDLE    ppbunAttrs;  /*  Default attribute values. */
BOOL       rc;          /*  Success indicator. */

rc = GpiSetDefAttrs(hps, lPrimType, flAttrMask,
       ppbunAttrs);
```

GpiSetDefAttrs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetDefAttrs Parameter - lPrimType

lPrimType (LONG) - input
Primitive type.

The primitive type for which default attributes are to be set:

PRIM_LINE
Line and arc primitives
PRIM_CHAR
Character primitives
PRIM_MARKER

	Marker primitives
PRIM_AREA	
	Area primitives
PRIM_IMAGE	
	Image primitives.

GpiSetDefAttrs Parameter - flAttrMask

flAttrMask (ULONG) - input
Attributes mask.

Each flag that is set indicates that the *ppbunAttrs* buffer contains data for the corresponding attribute. If all the flags in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

Line attributes:

LBB_COLOR	Line color
LBB_MIX_MODE	Line mix
LBB_WIDTH	Line width
LBB_GEOM_WIDTH	Geometric line width
LBB_TYPE	Line type
LBB_END	Line end
LBB_JOIN	Line join.

Character attributes:

CBB_COLOR	Character color
CBB_BACK_COLOR	Character background color
CBB_MIX_MODE	Character mix
CBB_BACK_MIX_MODE	Character background mix
CBB_SET	Character set
CBB_MODE	Character mode
CBB_BOX	Character box
CBB_ANGLE	Character angle
CBB_SHEAR	Character shear
CBB_DIRECTION	Character direction
CBB_EXTRA	Character extra
CBB_BREAK_EXTRA	Character break extra

Marker attributes:

MBB_COLOR	Marker color
MBB_BACK_COLOR	Marker background color
MBB_MIX_MODE	Marker mix
MBB_BACK_MIX_MODE	Marker background mix
MBB_SET	Marker set
MBB_SYMBOL	Marker symbol
MBB_BOX	Marker box.

Pattern attributes (areas):

ABB_COLOR	Area color
ABB_BACK_COLOR	Area background color
ABB_MIX_MODE	Area mix
ABB_BACK_MIX_MODE	Area background mix
ABB_SET	Pattern set
ABB_SYMBOL	Pattern symbol
ABB_REF_POINT	Pattern reference point.

Image attributes:

IBB_COLOR	Image color
IBB_BACK_COLOR	Image background color
IBB_MIX_MODE	Image mix
IBB_BACK_MIX_MODE	Image background mix.

GpiSetDefAttrs Parameter - ppbunAttrs

ppbunAttrs (PBUNDLE) - input
Default attribute values.

This is a structure containing default attribute values for each attribute for which the *flAttrMask* flag is set, at the correct offset as specified below for the particular primitive type.

Line attributes: *ppbunAttrs* consists of a LINEBUNDLE structure.

Character attributes: *ppbunAttrs* consists of a CHARBUNDLE structure.

Marker attributes: *ppbunAttrs* consists of a MARKERBUNDLE structure.

Pattern attributes (areas): *ppbunAttrs* consists of an AREABUNDLE structure.

Image attributes: *ppbunAttrs* consists of an IMAGEBUNDLE structure.

GpiSetDefAttrs Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetDefAttrs - Parameters

hps (HPS) - input
Presentation-space handle.

lPrimType (LONG) - input
Primitive type.

The primitive type for which default attributes are to be set:

PRIM_LINE
: Line and arc primitives

PRIM_CHAR
: Character primitives

PRIM_MARKER
: Marker primitives

PRIM_AREA
: Area primitives

PRIM_IMAGE
: Image primitives.

flAttrMask (ULONG) - input

Attributes mask.

Each flag that is set indicates that the *ppbunAttrs* buffer contains data for the corresponding attribute. If all the flags in *flAttrMask* are 0, the *ppbunAttrs* buffer address is not used.

Line attributes:

LBB_COLOR	Line color
LBB_MIX_MODE	Line mix
LBB_WIDTH	Line width
LBB_GEOM_WIDTH	Geometric line width
LBB_TYPE	Line type
LBB_END	Line end
LBB_JOIN	Line join.

Character attributes:

CBB_COLOR	Character color
CBB_BACK_COLOR	Character background color
CBB_MIX_MODE	Character mix
CBB_BACK_MIX_MODE	Character background mix
CBB_SET	Character set
CBB_MODE	Character mode
CBB_BOX	Character box
CBB_ANGLE	Character angle
CBB_SHEAR	Character shear
CBB_DIRECTION	Character direction
CBB_EXTRA	Character extra
CBB_BREAK_EXTRA	Character break extra

Marker attributes:

MBB_COLOR	Marker color
MBB_BACK_COLOR	Marker background color
MBB_MIX_MODE	Marker mix
MBB_BACK_MIX_MODE	Marker background mix
MBB_SET	Marker set
MBB_SYMBOL	Marker symbol
MBB_BOX	Marker box.

Pattern attributes (areas):

ABB_COLOR	Area color
ABB_BACK_COLOR	Area background color
ABB_MIX_MODE	Area mix
ABB_BACK_MIX_MODE	Area background mix
ABB_SET	Pattern set
ABB_SYMBOL	Pattern symbol
ABB_REF_POINT	Pattern reference point.

Image attributes:

IBB_COLOR	Image color
IBB_BACK_COLOR	Image background color
IBB_MIX_MODE	Image mix
IBB_BACK_MIX_MODE	Image background mix.

ppbunAttrs (PBUNDLE) - input
Default attribute values.

This is a structure containing default attribute values for each attribute for which the *flAttrMask* flag is set, at the correct offset as specified below for the particular primitive type.

Line attributes: *ppbunAttrs* consists of a LINEBUNDLE structure.

Character attributes: *ppbunAttrs* consists of a CHARBUNDLE structure.

Marker attributes: *ppbunAttrs* consists of a MARKERBUNDLE structure.

Pattern attributes (areas): *ppbunAttrs* consists of an AREABUNDLE structure.

Image attributes: *ppbunAttrs* consists of an IMAGEBUNDLE structure.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDefAttrs - Remarks

Attributes are reset to their default values at the following times:

- When the presentation space is associated with a device context (see GpiAssociate).
- When GpiResetPS is issued.
- When drawing of a chained segment begins or ends (see GpiOpenSegment and GpiCloseSegment for more details).
- When an attribute-setting function (for example, GpiSetAttrs) that sets an attribute to its default value is issued, or interpreted in a retained segment during a drawing operation.

Each attribute has an initial default value, established when the presentation space is first created. The value of this is given under the appropriate GpiSet.... call. The default value can be changed by GpiSetDefAttrs. Changing the default value takes effect immediately for the current value, if this is set to default at the time.

Each attribute of the primitive type in question is represented by one flag in the *flAttrMask* parameter. Any attribute for which the appropriate flag is set has its default value updated to the value specified in the *ppbunAttrs* structure. If the attribute is currently set to take the default value, it is immediately assigned the new default value. The default value of any attribute for which the appropriate flag in *flAttrMask* is not set is unchanged.

The data in the *ppbunAttrs* buffer consists of a structure of attribute data. The layout of the structure is fixed for each primitive type. Only data for attributes for which the flag is set in *flAttrMask* is inspected; any other data is ignored.

Note: The buffer need be no longer than is necessary to contain the data for the highest offset attribute referenced.

If an attempt is made to set an invalid default value by this function, none of the specified default attribute values is changed. Note, however, that some invalid default attribute values (for example, certain color and mix values) may not be detected until the attribute is set to default and used, at which point the implementation optionally defaults them, or causes an error to be logged.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information. Also, in a metafile, the default line width (see GpiSetLineWidth) is always rounded to an integer value, as is the default character box (see GpiSetCharBox) for GPIF_SHORT format presentation spaces (see GpiCreatePS).

GpiSetDefAttrs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PRIMITIVE_TYPE (0x20B9)
An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.

PMERR_UNSUPPORTED_ATTR (0x2109)

An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.

PMERR_INV_COLOR_ATTR (0x2053)
An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_BACKGROUND_COL_ATTR (0x2044)
An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_MIX_ATTR (0x20A3)
An invalid mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_LINE_WIDTH_ATTR (0x2096)
An invalid line width attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_GEOM_LINE_WIDTH_ATTR (0x2078)
An invalid geometric line width attribute value was specified.

PMERR_INV_LINE_TYPE_ATTR (0x2095)
An invalid line type attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_LINE_END_ATTR (0x2093)
An invalid line end attribute value was specified.

PMERR_INV_LINE_JOIN_ATTR (0x2094)
An invalid line join attribute value was specified.

PMERR_INV_CHAR_SET_ATTR (0x204F)
An invalid character setid attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_CHAR_MODE_ATTR (0x204D)
An invalid character mode attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_CHAR_DIRECTION_ATTR (0x204C)
An invalid character direction attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_CHAR_SHEAR_ATTR (0x2050)
An invalid character shear attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_CHAR_ANGLE_ATTR (0x204B)
The default character angle attribute value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_MARKER_SET_ATTR (0x2099)
An invalid marker set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_MARKER_SYMBOL_ATTR (0x209A)
An invalid marker symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_PATTERN_SET_ATTR (0x20B2)
An invalid pattern set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_PATTERN_ATTR (0x20B0)
An invalid pattern symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_UNSUPPORTED_ATTR_VALUE (0x210A)
An attribute value was specified with GpiSetAttrs that is not supported.

PMERR_INV_PATTERN_SET_FONT (0x20B3)

An attempt was made to use an unsuitable font as a pattern set.

PMERR_HUGE_FONTS_NOT_SUPPORTED (0x2035)
An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

GpiSetDefAttrs - Example Code

This function sets the default color of line and arc primitives to blue.

```
#define INCL_GPIDEFAULTS
#define INCL_GPIPRIMITIVES  /* for parameter definitions */
#include <OS2.H>
HPS     hps;    /* Presentation space handle     */
LINEBUNDLE bunAttrs; /* Information for color    */

bunAttrs.lColor = CLR_BLUE;


GpiSetDefAttrs(hps,
               PRIM_LINE,    /* line and arc primitives. */
               LBB_COLOR,    /* color information, which is */
                             /* contained in bunAttrs       */
               &bunAttrs);
```

GpiSetDefAttrs - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiSetDefaultViewMatrix

GpiSetDefaultViewMatrix - Syntax

This function sets the default viewing transform that is to apply to the whole picture.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
```

```
LONG         lCount;       /*  Number of elements. */
PMATRIXLF    pmatlfarray;  /*  Transformation matrix. */
LONG         lOptions;     /*  Transform options. */
BOOL         rc;           /*  Success indicator. */

rc = GpiSetDefaultViewMatrix(hps, lCount,
       pmatlfarray, lOptions);
```

GpiSetDefaultViewMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetDefaultViewMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements supplied in *pmatlfarray*, that are to be examined, starting from the beginning of the structure. If *lCount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. Specifying *lCount* = 0 means that the identity matrix is used.

GpiSetDefaultViewMatrix Parameter - pmatlfarray

pmatlfarray (PMATRIXLF) - input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation-space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

GpiSetDefaultViewMatrix Parameter - lOptions

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the *pmatlfarray* parameter should be used to modify the existing default viewing transform.

Possible values are:

TRANSFORM_REPLACE
The previous default viewing transform is discarded and replaced by the specified transform.
TRANSFORM_ADD
The specified transform is combined with the existing default viewing transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms.
TRANSFORM_PREEMPT
The specified transform is combined with the existing default viewing transform, in the order (1) new transform, (2) existing transform.

GpiSetDefaultViewMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDefaultViewMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of elements.

The number of elements supplied in *pmatlfarray*, that are to be examined, starting from the beginning of the structure. If *lCount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. Specifying *lCount* = 0 means that the identity matrix is used.

pmatlfarray (PMATRIXLF) - input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation-space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the *pmatlfarray* parameter should be used to modify the existing default viewing transform.

Possible values are:

TRANSFORM_REPLACE
The previous default viewing transform is discarded and replaced by the specified transform.
TRANSFORM_ADD
The specified transform is combined with the existing default viewing transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms.
TRANSFORM_PREEMPT
The specified transform is combined with the existing default viewing transform, in the order (1) new transform, (2) existing transform.

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetDefaultViewMatrix - Remarks

The transform matrix specified is used to update any previous default viewing transform, depending upon the value of *lOptions* .

The transform is specified as a one-dimensional array of *lCount* elements, being the first n elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

```
Matrix                  Array


a  b  0
c  d  0          (a,b,0,c,d,0,e,f,1)
e  f  1
```

The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point:

```
(a*x + c*y + e, b*x + d*y + f)
```

The initial value of the default viewing transform is the identity matrix, as shown below:

```
Matrix                  Array


1  0  0
0  1  0          (1,0,0,0,1,0,0,0,1)
0  0  1
```

If scaling values greater than unity are given (which only applies if the presentation space coordinate format, as set by the GpiCreatePS function, is GPIF_LONG), it is possible for the combined effect of this and any other relevant transforms to exceed fixed-point implementation limits. This causes an error.

This function must not be issued in a path or area bracket.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetDefaultViewMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_TRANSFORM_TYPE (0x20D0)
An invalid options parameter was specified with a transform matrix function.

PMERR_INV_MATRIX_ELEMENT (0x209B)
An invalid transformation matrix element was specified.

GpiSetDefaultViewMatrix - Related Functions

Related Functions

- GpiQueryDefaultViewMatrix
- GpiSetViewingTransformMatrix

GpiSetDefaultViewMatrix - Example Code

This example uses the GpiSetDefaultViewMatrix function to replace the existing default viewing transformation. The new transformation translates drawing to the right by 100 units.

```
#define INCL_GPITRANSFORMS      /* Transform functions          */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */
/* transform matrix */
MATRIXLF matlf = {MAKEFIXED(1,0), 0, 0, 0, MAKEFIXED(1,0), 0, 100};

fSuccess = GpiSetDefaultViewMatrix(hps, 7L, &matlf,
                                   TRANSFORM_REPLACE);
```

GpiSetDefaultViewMatrix - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetDefTag

GpiSetDefTag - Syntax

This function specifies the default value of the primitive tag (see GpiSetTag).

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
LONG    lTag;  /*  Default tag identifier. */
BOOL    rc;    /*  Success indicator. */

rc = GpiSetDefTag(hps, lTag);
```

GpiSetDefTag Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetDefTag Parameter - lTag

lTag (LONG) - input
Default tag identifier.

GpiSetDefTag Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDefTag - Parameters

hps (HPS) - input
Presentation-space handle.

lTag (LONG) - input
Default tag identifier.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDefTag - Remarks

The primitive tag is reset to its default value at the following times:

- When the presentation space is associated with a device context (see GpiAssociate).
- When GpiResetPS is issued.
- When drawing of a chained segment begins or ends (see GpiOpenSegment and GpiCloseSegment for more details).

The initial default value of the primitive tag, when the presentation space is first created, is 0. The default value can be changed by GpiSetDefTag. Changing the default value has an immediate effect on the current primitive tag, if this is currently set to the default value.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetDefTag - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiSetDefTag - Related Functions

Related Functions

- GpiQueryDefAttrs
- GpiQueryTag
- GpiSetTag

GpiSetDefTag - Example Code

This function specifies the default value of the primitive tag (see GpiSetTag).

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>


HPS     hps;    /* Presentation space handle     */

GpiSetDefTag(hps, 1L);
```

GpiSetDefTag - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetDefViewingLimits

GpiSetDefViewingLimits - Syntax

This function specifies the default value of the viewing limits (see GpiSetViewingLimits).

```
#define INCL_GPIDEFAULTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;         /*  Presentation-space handle. */
PRECTL    prclLimits;  /*  Default viewing limits. */
BOOL      rc;          /*  Success indicator. */

rc = GpiSetDefViewingLimits(hps, prclLimits);
```

GpiSetDefViewingLimits Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetDefViewingLimits Parameter - prclLimits

prclLimits (PRECTL) - input
Default viewing limits.

GpiSetDefViewingLimits Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDefViewingLimits - Parameters

hps (HPS) - input
Presentation-space handle.

prclLimits (PRECTL) - input
Default viewing limits.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDefViewingLimits - Remarks

The viewing limits are reset to their default value at the following times:

- When the presentation space is associated with a device context (see GpiAssociate).
- When GpiResetPS is issued.
- When drawing of a chained segment begins or ends (see GpiOpenSegment and GpiCloseSegment for more details).

The initial default value of the viewing limits, when the presentation space is first created, is no clipping. The default value can be changed by GpiSetDefViewingLimits. Changing the default values has an immediate effect on the current viewing limits, if these are currently set to the default value.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the

Presentation Manager Programming Reference for more information.

GpiSetDefViewingLimits - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetDefViewingLimits - Related Functions

Related Functions

- GpiQueryDefViewingLimits
- GpiQueryGraphicsField
- GpiQueryViewingLimits
- GpiSetGraphicsField
- GpiSetViewingLimits

GpiSetDefViewingLimits - Example Code

In this example the default model space clipping region width is set to 100.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */
RECTL rclLimits; /* viewing limits.    */


rclLimits.xRight = 100;
rclLimits.xLeft = 100;

GpiSetDefViewingLimits(hps,
                       &rclLimits);
```

GpiSetDefViewingLimits - Topics

Select an item:
Syntax

Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetDrawControl

GpiSetDrawControl - Syntax

This function sets options for subsequent drawing operations.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lControl;  /*  Drawing control. */
LONG    lValue;    /*  Required value of the drawing control. */
BOOL    rc;        /*  Success indicator. */

rc = GpiSetDrawControl(hps, lControl, lValue);
```

GpiSetDrawControl Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetDrawControl Parameter - lControl

lControl (LONG) - input
Drawing control.

Note: Controls identified by an asterisk (*) are the only ones relevant to a micro-presentation space. Any other control settings are ignored for a micro-presentation space.

DCTL_ERASE
Erase before draw. Perform an implicit GpiErase operation before GpiDrawChain, GpiDrawFrom, or GpiDrawSegment. The output display area of the Device Context associated with the specified presentation space is cleared before drawing.

DCTL_DISPLAY (*)
: Enable drawing to occur on the output medium. If this control is set to off, then except for GpiErase, no output operations appear on the output medium. This includes raster operations, such as drawing primitives, and GpiDraw... operations.

DCTL_BOUNDARY (*)
: Accumulate boundary data. During any output operations except GpiErase, accumulate the bounding rectangle of the drawing.

DCTL_DYNAMIC
: Draw dynamic segments. Perform an implicit GpiRemoveDynamics before GpiDrawChain, GpiDrawFrom, or GpiDrawSegment, and an implicit GpiDrawDynamics afterwards.

 Note that, to either remove or draw dynamic segments, the system forces the foreground mix to FM_XOR, and the background mix to BM_LEAVEALONE. If the first nondynamic segment being drawn (immediately after the dynamic segments have been removed) has the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), it may be necessary for it to set the mix modes itself before drawing. Similar considerations might apply for any subsequent drawing after the dynamic segments have been replaced.

DCTL_CORRELATE (*)
: If this control is set, any GpiPutData, GpiElement, GpiPlayMetaFile, or individual drawing primitives which are passed across the API outside a segment bracket cause a correlation operation to be performed, and a return code to be set if a hit occurs. (Correlation inside segments, both retained and nonretained, is controlled by the segment attribute ATTR_DETECTABLE).

 This control has an effect only in **draw** or **draw-and-retain** modes (see GpiSetDrawingMode).

GpiSetDrawControl Parameter - lValue

lValue (LONG) - input
: Required value of the drawing control.

DCTL_OFF
: Set control off

DCTL_ON
: Set control on.

GpiSetDrawControl Return Value - rc

rc (BOOL) - returns
: Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetDrawControl - Parameters

hps (HPS) - input
: Presentation-space handle.

lControl (LONG) - input
Drawing control.

Note: Controls identified by an asterisk (*) are the only ones relevant to a micro-presentation space. Any other control settings are ignored for a micro-presentation space.

DCTL_ERASE
Erase before draw. Perform an implicit GpiErase operation before GpiDrawChain, GpiDrawFrom, or GpiDrawSegment. The output display area of the Device Context associated with the specified presentation space is cleared before drawing.

DCTL_DISPLAY (*)
Enable drawing to occur on the output medium. If this control is set to off, then except for GpiErase, no output operations appear on the output medium. This includes raster operations, such as drawing primitives, and GpiDraw... operations.

DCTL_BOUNDARY (*)
Accumulate boundary data. During any output operations except GpiErase, accumulate the bounding rectangle of the drawing.

DCTL_DYNAMIC
Draw dynamic segments. Perform an implicit GpiRemoveDynamics before GpiDrawChain, GpiDrawFrom, or GpiDrawSegment, and an implicit GpiDrawDynamics afterwards.

Note that, to either remove or draw dynamic segments, the system forces the foreground mix to FM_XOR, and the background mix to BM_LEAVEALONE. If the first nondynamic segment being drawn (immediately after the dynamic segments have been removed) has the ATTR_FASTCHAIN attribute (see GpiSetInitialSegmentAttrs), it may be necessary for it to set the mix modes itself before drawing. Similar considerations might apply for any subsequent drawing after the dynamic segments have been replaced.

DCTL_CORRELATE (*)
If this control is set, any GpiPutData, GpiElement, GpiPlayMetaFile, or individual drawing primitives which are passed across the API outside a segment bracket cause a correlation operation to be performed, and a return code to be set if a hit occurs. (Correlation inside segments, both retained and nonretained, is controlled by the segment attribute ATTR_DETECTABLE).

This control has an effect only in **draw** or **draw-and-retain** modes (see GpiSetDrawingMode).

lValue (LONG) - input
Required value of the drawing control.

DCTL_OFF
Set control off
DCTL_ON
Set control on.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDrawControl - Remarks

The default value is DCTL_OFF for all controls except DCTL_DISPLAY (*). Its default value is DCTL_ON.

This function must not be issued in any of these cases:

- Inside an open segment
- Outside an open segment, but inside one of:
 - Area bracket
 - Element bracket

- Path bracket.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetDrawControl - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_DRAW_CONTROL (0x2063)
An invalid control parameter was specified with GpiSetDrawControl or GpiQueryDrawControl.

PMERR_INV_DRAW_VALUE (0x2064)
An invalid value parameter was specified with GpiSetDrawControl.

PMERR_INV_IN_SEG (0x208D)
An attempt was made to issue a function invalid inside a segment bracket.

PMERR_INV_IN_AREA (0x2085)
An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

PMERR_INV_IN_PATH (0x208B)
An attempt was made to issue a function invalid inside a path bracket.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

PMERR_INV_MICROPS_DRAW_CONTROL (0x20A0)
A draw control parameter was specified with GpiSetDrawControl that is invalid in a micro presentation space.

GpiSetDrawControl - Related Functions

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiErase
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetDrawingMode
- GpiSetStopDraw

GpiSetDrawControl - Example Code

The "display" drawing control is switched off, and the "accumulate-boundary-data" control is switched on.

```
#define INCL_GPICONTROL
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

  GpiResetBoundaryData(hps);
  GpiSetDrawControl(hps, DCTL_DISPLAY, DCTL_OFF);
```

GpiSetDrawControl - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetDrawingMode

GpiSetDrawingMode - Syntax

This function sets the drawing mode to control the handling of subsequent individual drawing primitive and attribute calls.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lMode;  /*  Mode to be used for subsequent drawing calls. */
BOOL    rc;     /*  Success indicator. */

rc = GpiSetDrawingMode(hps, lMode);
```

GpiSetDrawingMode Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetDrawingMode Parameter - lMode

lMode (LONG) - input
Mode to be used for subsequent drawing calls.

This parameter can contain a combination of the following values:

DM_DRAW
Draw, unless in an unchained segment
DM_RETAIN
Retain, if within a segment
DM_DRAWANDRETAIN
Draw-and-retain, combination of above.

GpiSetDrawingMode Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDrawingMode - Parameters

hps (HPS) - input
Presentation-space handle.

lMode (LONG) - input
Mode to be used for subsequent drawing calls.

This parameter can contain a combination of the following values:

DM_DRAW
Draw, unless in an unchained segment
DM_RETAIN
Retain, if within a segment
DM_DRAWANDRETAIN
Draw-and-retain, combination of above.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetDrawingMode - Remarks

The drawing mode affects the handling of subsequent individual drawing primitive and attribute calls, and the GpiPutData, GpiElement, and GpiPlayMetaFile functions.

Primitives and attributes can be drawn immediately, retained, or both, in the current segment.

Note: Any primitive and attribute setting calls that occur **outside** a segment (that is, outside a GpiOpenSegment - GpiCloseSegment bracket) are always treated as nonretained. Conversely, any segments that are not chained are always retained. This table summarizes how the actual drawing mode is arrived at:

Parameter	Context		
	Chained Segment	Unchained Segment	Outside Segment
DM_DRAWANDRETAIN	draw-and-retain	retain	draw
DM_RETAIN	retain	retain	draw
DM_DRAW	draw	retain	draw

The actual drawing mode (referred to when describing other Gpi calls) therefore depends upon the mode as set by GpiSetDrawingMode, together with the context, as in the table. It is this actual drawing mode that determines whether a drawing call is retained (**retain** or **draw-and-retain**), and whether it is drawn immediately (**draw** or **draw-and-retain**).

It is an error to try to set the drawing mode within a segment bracket, and also outside a segment bracket, if in one of the following:

- Area bracket
- Element bracket
- Path bracket.

The default drawing mode is DM_DRAW.

GpiSetDrawingMode - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_IN_AREA (0x2085)
An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

PMERR_INV_IN_PATH (0x208B)
An attempt was made to issue a function invalid inside a path bracket.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

PMERR_INV_IN_SEG (0x208D)

An attempt was made to issue a function invalid inside a segment bracket.

PMERR_INV_DRAWING_MODE (0x2065)
An invalid mode parameter was specified with GpiSetDrawControl not **draw-and-retain** or **draw**.

GpiSetDrawingMode - Related Functions

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiErase
- GpiGetData
- GpiOpenSegment
- GpiPutData
- GpiQueryDrawControl
- GpiQueryDrawingMode
- GpiQueryStopDraw
- GpiRemoveDynamics
- GpiSetStopDraw

GpiSetDrawingMode - Example Code

This example calls GpiSetDrawingMode to set the drawing mode to DRAW.

```
#define INCL_GPICONTROL         /* GPI control Functions        */
#include <os2.h>

BOOL  fSuccess;         /* success indicator                    */
HPS    hps;             /* Presentation-space handle            */

fSuccess = GpiSetDrawingMode(hps, DM_DRAW);
```

GpiSetDrawingMode - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetEditMode

GpiSetEditMode - Syntax

This function sets the current editing mode.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;    /*  Presentation-space handle. */
LONG    lMode;  /*  Edit mode. */
BOOL    rc;     /*  Success indicator. */

rc = GpiSetEditMode(hps, lMode);
```

GpiSetEditMode Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetEditMode Parameter - lMode

lMode (LONG) - input
Edit mode.

SEGEM_INSERT
Insert mode
SEGEM_REPLACE
Replace mode.

GpiSetEditMode Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetEditMode - Parameters

hps (HPS) - input
Presentation-space handle.

lMode (LONG) - input
Edit mode.

SEGEM_INSERT
Insert mode
SEGEM_REPLACE
Replace mode.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetEditMode - Remarks

This function determines whether a new element is to be inserted into a segment, moving any subsequent elements further along the segment, or whether a new element is to replace the current element.

In SEGEM_INSERT mode, when an element is generated, it is inserted following the element indicated by the element pointer. The element pointer is updated to point to the new element.

In SEGEM_REPLACE mode, when an element is generated, it replaces the element indicated by the element pointer. The element pointer does not change. It is an error if a new element is generated in SEGEM_REPLACE mode if the element pointer is 0 (as it is when a segment is opened).

The editing mode can be changed at any time, (except while within an element bracket), and is not an attribute of a specific segment. It only applies to the storing of data within retained segments. It is not an error to issue this function in other drawing modes; the value of the edit mode is set irrespective of the value of the draw mode.

This function is invalid within an element bracket. The default editing mode (set by GpiCreatePS or GpiResetPS) is SEGEM_INSERT.

GpiSetEditMode - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_EDIT_MODE (0x2069)
An invalid mode parameter was specified with GpiSetEditMode.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

PMERR_INV_MICROPS_FUNCTION (0x20A1)

An attempt was made to issue a function that is invalid in a micro presentation space.

GpiSetEditMode - Related Functions

Related Functions

- GpiCreatePS
- GpiOpenSegment
- GpiQueryEditMode

GpiSetEditMode - Example Code

This example sets the current editing mode to insert.

```
#define INCL_GPISEGEDITING
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */


GpiSetEditMode(hps, SEGEM_INSERT); /* insert mode. */
```

GpiSetEditMode - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetElementPointer

GpiSetElementPointer - Syntax

This function sets the element pointer, within the current segment, to the element number specified.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lElement;  /*  The element number required. */
BOOL    rc;        /*  Success indicator. */

rc = GpiSetElementPointer(hps, lElement);
```

GpiSetElementPointer Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetElementPointer Parameter - lElement

lElement (LONG) - input
The element number required.

If the value specified is negative, the element pointer is set to 0.

If the value specified is greater than the number of elements in the segment, the element pointer is set to the last element.

GpiSetElementPointer Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetElementPointer - Parameters

hps (HPS) - input
Presentation-space handle.

lElement (LONG) - input
The element number required.

If the value specified is negative, the element pointer is set to 0.

If the value specified is greater than the number of elements in the segment, the element pointer is set to the last element.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetElementPointer - Remarks

The currently open segment has an element pointer that points to a particular element in the segment; each element is placed into the segment at the place indicated by the pointer. When a retained segment is first opened, the element pointer is set to 0 (empty segment). It is incremented each time a call causes an element (a single API call) to be placed in the segment. When a segment is reopened, the element pointer is reset to 0 (that is, before the first element).

The element pointer for a segment is not remembered if the segment is closed and subsequently reopened.

This function is only valid when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently in progress. It is invalid within an element bracket.

GpiSetElementPointer - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_NO_CURRENT_SEG (0x20E6)
An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_IN_ELEMENT (0x2089)
An attempt was made to issue a function invalid inside an element bracket.

GpiSetElementPointer - Related Functions

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement

- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointerAtLabel

GpiSetElementPointer - Example Code

This function sets the element pointer, within the current segment, to 0.

```
#define INCL_GPICONTROL
#define INCL_GPISEGEDITING
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */


 /* This example uses the GpiSetElementPointer function to move  */
 /* the element pointer to an element to be edited.              */

 GpiSetDrawingMode(hps, DM_RETAIN);/* set DM_RETAIN drawing mode */
 GpiOpenSegment(hps, 2L);               /* open segment to edit */
 GpiSetElementPointer(hps, 3L);         /* move element pointer
                                           to 3rd element       */
 GpiSetColor(hps,CLR_GREEN);          /* new element changes
                                           color to green       */
 GpiCloseSegment(hps);                  /* close the segment    */
```

GpiSetElementPointer - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetElementPointerAtLabel

GpiSetElementPointerAtLabel - Syntax

This function sets the element pointer, within the current segment, to the element containing the specified label.

```
#define INCL_GPISEGEDITING /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lLabel;  /*  Required label. */
BOOL    rc;      /*  Success indicator. */

rc = GpiSetElementPointerAtLabel(hps, lLabel);
```

GpiSetElementPointerAtLabel Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetElementPointerAtLabel Parameter - lLabel

lLabel (LONG) - input
Required label.

GpiSetElementPointerAtLabel Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetElementPointerAtLabel - Parameters

hps (HPS) - input
Presentation-space handle.

lLabel (LONG) - input
Required label.

rc (BOOL) - returns
Success indicator.

TRUE

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetElementPointerAtLabel - Remarks

The search starts from the element pointed to by the current element pointer. If the specified label is not found between there and the end of the segment, an error is generated and the element pointer is left unchanged. (Also see GpiSetElementPointer.)

This function is valid only when the drawing mode (see GpiSetDrawingMode) is set to **retain** (not **draw-and-retain**), and a segment bracket is currently open. It is not valid within an element bracket.

GpiSetElementPointerAtLabel - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
: An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

PMERR_NO_CURRENT_SEG (0x20E6)
: An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.

PMERR_INV_IN_ELEMENT (0x2089)
: An attempt was made to issue a function invalid inside an element bracket.

PMERR_LABEL_NOT_FOUND (0x20D6)
: The specified element label did not exist.

GpiSetElementPointerAtLabel - Related Functions

Related Functions

- GpiBeginElement
- GpiDeleteElement
- GpiDeleteElementRange
- GpiDeleteElementsBetweenLabels
- GpiElement
- GpiEndElement
- GpiLabel
- GpiOffsetElementPointer
- GpiQueryElement
- GpiQueryElementPointer
- GpiQueryElementType
- GpiSetElementPointer

GpiSetElementPointerAtLabel - Example Code

This function sets the element pointer at label 1.

```
#define INCL_GPISEGEDITING
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

GpiSetElementPointerAtLabel(hps, 1L);
```

GpiSetElementPointerAtLabel - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetGraphicsField

GpiSetGraphicsField - Syntax

This function sets the size and position of the graphics field in presentation page units.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
PRECTL    prclField;  /*  Graphics field. */
BOOL      rc;         /*  Success indicator. */

rc = GpiSetGraphicsField(hps, prclField);
```

GpiSetGraphicsField Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetGraphicsField Parameter - prclField

prclField (PRECTL) - input
Graphics field.

It is an error if the top coordinate is less than the bottom, or the right coordinate is less than the left.

All values are in presentation-page units.

GpiSetGraphicsField Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetGraphicsField - Parameters

hps (HPS) - input
Presentation-space handle.

prclField (PRECTL) - input
Graphics field.

It is an error if the top coordinate is less than the bottom, or the right coordinate is less than the left.

All values are in presentation-page units.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetGraphicsField - Remarks

The graphics field specifies a clipping boundary within the presentation page.

The boundaries are inclusive, so that points on them are not clipped (removed). By default, no clipping is performed.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetGraphicsField - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_GRAPHICS_FIELD (0x207A)
An invalid field parameter was specified with GpiSetGraphicsField.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetGraphicsField - Related Functions

Related Functions

- GpiExcludeClipRectangle
- GpiIntersectClipRectangle
- GpiOffsetClipRegion
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiQueryDefViewingLimits
- GpiQueryGraphicsField
- GpiQueryViewingLimits
- GpiSetClipPath
- GpiSetClipRegion
- GpiSetDefViewingLimits
- GpiSetViewingLimits

GpiSetGraphicsField - Example Code

This example sets the graphics field to 400x400 with the left bottom corner at (25,25).

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

RECTL rclField = {25, /* x coordinate of left-hand edge of */
                      /* rectangle. */
                  25, /* y coordinate of bottom edge of    */
```

```
                    /* rectangle. */
                425,/* x coordinate of right-hand edge of */
                    /* rectangle. */
                425};/* y coordinate of top edge of rectangle. */

GpiSetGraphicsField(hps, &rclField);
```

GpiSetGraphicsField - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetInitialSegmentAttrs

GpiSetInitialSegmentAttrs - Syntax

This function specifies a segment attribute that is used when a segment is subsequently created.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;         /*  Presentation-space handle. */
LONG    lAttribute;  /*  Segment attribute. */
LONG    lValue;      /*  Attribute value. */
BOOL    rc;          /*  Success indicator. */

rc = GpiSetInitialSegmentAttrs(hps, lAttribute,
       lValue);
```

GpiSetInitialSegmentAttrs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetInitialSegmentAttrs Parameter - lAttribute

lAttribute (LONG) - input
Segment attribute.

The following values can be ORed together to generate this parameter:

ATTR_DETECTABLE
Detectability.

This can be used to determine whether a correlation function can be performed on the primitives within the segment. For correlation on retained segments see:

- GpiCorrelateChain
- GpiCorrelateFrom
- GpiCorrelateSegment. Correlation on primitives outside segments is controlled by the correlate flag on draw controls (see GpiSetDrawControl).

ATTR_VISIBLE
Visibility.

Controls whether a segment is to be made visible on the output medium.

ATTR_CHAINED
Chained.

Controls whether the segment is a root segment to be included in the segment drawing chain. In **draw** or **draw-and-retain** modes (see GpiSetDrawingMode) a chained segment is drawn as it passes across the API; an unchained segment is not.

Unchained segments are usually called from another segment. They can also be segments that are inserted into the chain later (with GpiSetSegmentPriority or GpiSetSegmentAttrs), or segments that are drawn individually with GpiDrawSegment.

ATTR_DYNAMIC
Dynamic.

Controls whether the segment is to be dynamic; that is, drawn using exclusive-OR, so that it can be readily erased by redrawing it. (See GpiDrawDynamics, GpiRemoveDynamics, and the DCTL_DYNAMIC option of GpiSetDrawControl.)

Only retained segments can be dynamic.

The dynamic segment attribute is always ignored if the segment is not currently chained.

ATTR_FASTCHAIN
Fast chaining.

Controls whether, for a chained segment, the system can assume that all primitive attributes need not be reset to default values before execution of the segment.

ATTR_PROP_DETECTABLE
Propagate detectability.

Controls whether the value of the detectability attribute for a segment should be propagated (forced) to all segments beneath it in the hierarchy.

ATTR_PROP_VISIBLE
Propagate visibility.

Controls whether the value of the visibility attribute for a segment should be propagated (forced) to all segments beneath it in the hierarchy.

GpiSetInitialSegmentAttrs Parameter - lValue

lValue (LONG) - input
Attribute value.

ATTR_ON
On/yes
ATTR_OFF
Off/no.

GpiSetInitialSegmentAttrs Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetInitialSegmentAttrs - Parameters

hps (HPS) - input
Presentation-space handle.

lAttribute (LONG) - input
Segment attribute.

The following values can be ORed together to generate this parameter:

ATTR_DETECTABLE
Detectability.

This can be used to determine whether a correlation function can be performed on the primitives within the segment. For correlation on retained segments see:

- GpiCorrelateChain
- GpiCorrelateFrom
- GpiCorrelateSegment. Correlation on primitives outside segments is controlled by the correlate flag on draw controls (see GpiSetDrawControl).

ATTR_VISIBLE
Visibility.

Controls whether a segment is to be made visible on the output medium.

ATTR_CHAINED
Chained.

Controls whether the segment is a root segment to be included in the segment drawing chain. In **draw** or **draw-and-retain** modes (see GpiSetDrawingMode) a chained segment is drawn as it passes across the API; an unchained segment is not.

Unchained segments are usually called from another segment. They can also be segments that are inserted into the chain later (with GpiSetSegmentPriority or GpiSetSegmentAttrs), or segments that are drawn individually with GpiDrawSegment.

ATTR_DYNAMIC
Dynamic.

Controls whether the segment is to be dynamic; that is, drawn using exclusive-OR, so that it can be readily erased by redrawing it. (See GpiDrawDynamics, GpiRemoveDynamics, and the DCTL_DYNAMIC option of GpiSetDrawControl.)

Only retained segments can be dynamic.

The dynamic segment attribute is always ignored if the segment is not currently chained.

ATTR_FASTCHAIN
Fast chaining.

Controls whether, for a chained segment, the system can assume that all primitive attributes need not be reset to default values before execution of the segment.

ATTR_PROP_DETECTABLE
Propagate detectability.

Controls whether the value of the detectability attribute for a segment should be propagated (forced) to all segments beneath it in the hierarchy.

ATTR_PROP_VISIBLE
Propagate visibility.

Controls whether the value of the visibility attribute for a segment should be propagated (forced) to all segments beneath it in the hierarchy.

lValue (LONG) - input
Attribute value.

ATTR_ON
On/yes
ATTR_OFF
Off/no.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetInitialSegmentAttrs - Remarks

Initial segment attributes are modal settings used to determine the initial attributes of new segments as they are created; that is, when an GpiOpenSegment function is issued, and the segment does not already exist. The default values of initial segment attributes are:

- Not detectable
- Visible
- Chained
- Not dynamic
- Fast chaining
- Propagate detectability
- Propagate visibility.

A nonretained segment can never be given the **dynamic** attribute.

Primitives outside segments are not affected by GpiSetInitialSegmentAttrs.

GpiSetInitialSegmentAttrs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_ATTR (0x20C5)
An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.

PMERR_INV_SEG_ATTR_VALUE (0x20C6)
An invalid attribute value parameter was specified with GpiSetSegmentAttrs or GpiSetInitialSegmentAttrs.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiSetInitialSegmentAttrs - Related Functions

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiQueryInitialSegmentAttrs
- GpiSetSegmentAttrs
- GpiSetSegmentPriority

GpiSetInitialSegmentAttrs - Example Code

This function specifies a segment attribute that is used when a segment is subsequently created. In this example, the most common attributes are selected.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */


GpiSetInitialSegmentAttrs (hps,
                           ATTR_DETECTABLE |
                           ATTR_VISIBLE |
                           ATTR_DYNAMIC |
                           ATTR_PROP_DETECTABLE |
                           ATTR_PROP_VISIBLE,
                           ATTR_ON);
```

GpiSetInitialSegmentAttrs - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetLineEnd

GpiSetLineEnd - Syntax

This function sets the current line-end attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lLineEnd;  /*  Style of line end. */
BOOL    rc;        /*  Success indicator. */

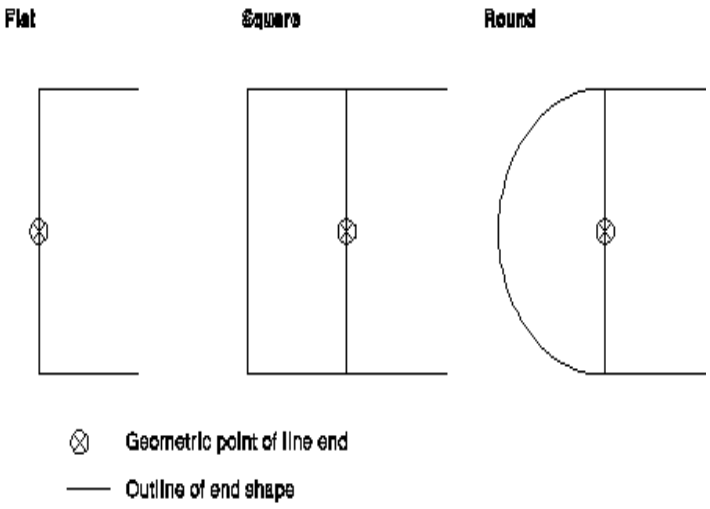
rc = GpiSetLineEnd(hps, lLineEnd);
```

GpiSetLineEnd Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetLineEnd Parameter - lLineEnd

lLineEnd (LONG) - input
Style of line end.



The possible values for this parameter are:

LINEEND_DEFAULT
: Use default, same as LINEEND_FLAT (unless changed with GpiSetDefAttrs)

LINEEND_FLAT
: Flat

LINEEND_SQUARE
: Square

LINEEND_ROUND
: Round.

GpiSetLineEnd Return Value - rc

rc (BOOL) - returns
Success indicator.

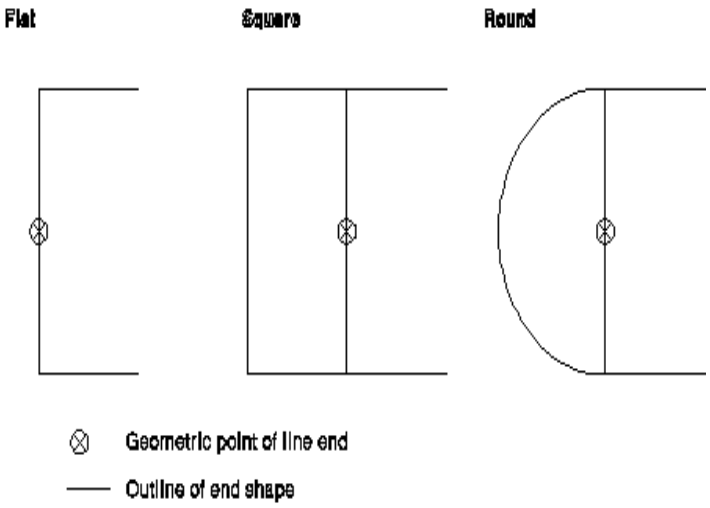
TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetLineEnd - Parameters

hps (HPS) - input
Presentation-space handle.

lLineEnd (LONG) - input
Style of line end.



The possible values for this parameter are:

LINEEND_DEFAULT
: Use default, same as LINEEND_FLAT (unless changed with GpiSetDefAttrs)

LINEEND_FLAT
: Flat

LINEEND_SQUARE
: Square

LINEEND_ROUND
: Round.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetLineEnd - Remarks

The line-end attribute defines the shape of the ends of lines or arcs at the beginning and end of an open figure. This attribute is used only in the GpiModifyPath function (with a *lMode* parameter of MPATH_STROKE) or in the GpiStrokePath function.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the line-end attribute is preserved.

GpiSetLineEnd - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)

An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LINE_END_ATTR (0x2093)
An invalid line end attribute value was specified.

GpiSetLineEnd - Related Functions

Related Functions

- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom

GpiSetLineEnd - Graphic Elements and Orders

Element Type: **OCODE_GSLE**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Line End

Element Type: **OCODE_GPSLE**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Line End

GpiSetLineEnd - Example Code

This function sets the line end to be square (as opposed to round for example).

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

GpiSetLineEnd(hps,
              LINEEND_SQUARE);
```

GpiSetLineEnd - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetLineJoin

GpiSetLineJoin - Syntax

This function sets the current line-join attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;        /*  Presentation-space handle. */
LONG    lLineJoin;  /*  Style of line join. */
BOOL    rc;         /*  Success indicator. */

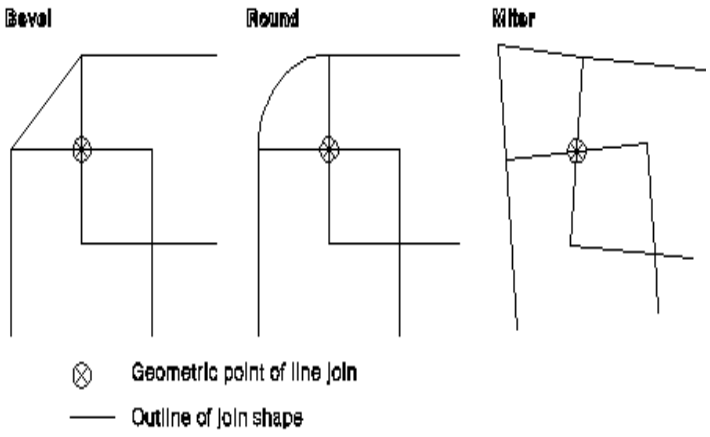
rc = GpiSetLineJoin(hps, lLineJoin);
```

GpiSetLineJoin Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetLineJoin Parameter - lLineJoin

lLineJoin (LONG) - input
Style of line join.



The possible values for this parameter are:

LINEJOIN_DEFAULT
: Use default, same as LINEJOIN_BEVEL (unless changed with GpiSetDefAttrs)

LINEJOIN_BEVEL
: Bevel

LINEJOIN_ROUND
: Round

LINEJOIN_MITRE
: Miter.

GpiSetLineJoin Return Value - rc

rc (BOOL) - returns
Success indicator.

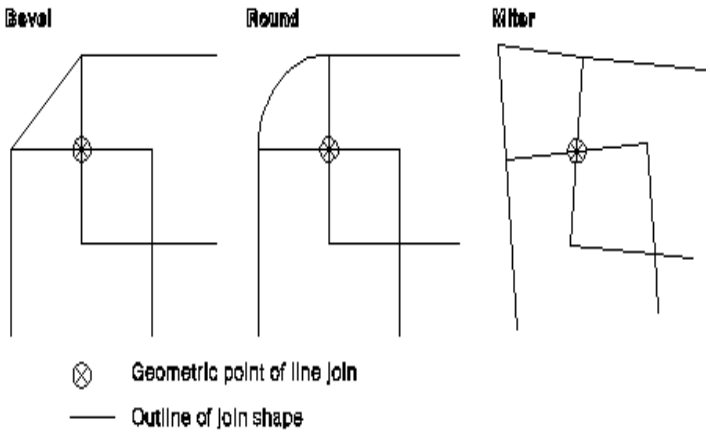
TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetLineJoin - Parameters

hps (HPS) - input
Presentation-space handle.

lLineJoin (LONG) - input
Style of line join.



The possible values for this parameter are:

LINEJOIN_DEFAULT
: Use default, same as LINEJOIN_BEVEL (unless changed with GpiSetDefAttrs)

LINEJOIN_BEVEL
: Bevel

LINEJOIN_ROUND
: Round

LINEJOIN_MITRE
: Miter.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetLineJoin - Remarks

The line-join attribute defines how individual lines and arcs within a figure are joined together. This attribute is used only during a GpiModifyPath function (with a *lMode* parameter of MPATH_STROKE) or a GpiStrokePath function.

For LINEJOIN_MITRE, where the lines going into a join are nearly parallel (a very sharp change in direction), a miter join could potentially extend to a distance that approaches infinity. To prevent this, whenever the ratio of the miter length to the geometric line width exceeds 10, a bevel join is drawn instead. (The miter length is the distance from the point at which the inner edges of the wideline intersect, to the point at which the outer edges of the wideline intersect.)

The attribute mode (see GpiSetAttrMode) determines whether the current value of the line-join attribute is preserved.

GpiSetLineJoin - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LINE_JOIN_ATTR (0x2094)
An invalid line join attribute value was specified.

GpiSetLineJoin - Related Functions

Related Functions

- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom

GpiSetLineJoin - Graphic Elements and Orders

Element Type: **OCODE_GSLJ**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Line Join

Element Type: **OCODE_GPSLJ**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Line Join

GpiSetLineJoin - Example Code

This function sets the line-join to be round (as opposed to bevel or miter).

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

GpiSetLineEnd(hps,
```

```
            LINEJOIN_ROUND);
```

GpiSetLineJoin - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetLineType

GpiSetLineType - Syntax

This function sets the current cosmetic line-type attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;        /*  Presentation-space handle. */
LONG    lLineType;  /*  Line types available. */
BOOL    rc;         /*  Success indicator. */

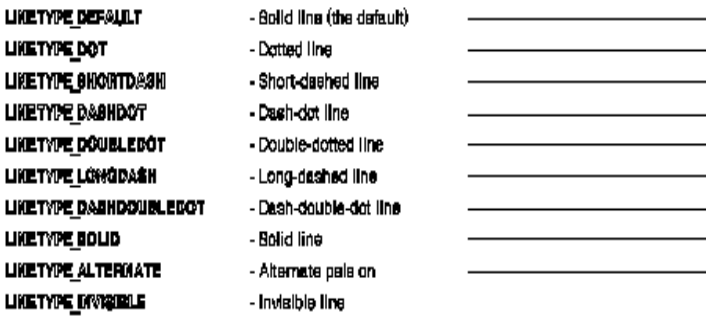
rc = GpiSetLineType(hps, lLineType);
```

GpiSetLineType Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetLineType Parameter - lLineType

lLineType (LONG) - input
Line types available.



GpiSetLineType Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetLineType - Parameters

hps (HPS) - input
Presentation-space handle.

lLineType (LONG) - input
Line types available.

LINETYPE_DEFAULT	- Solid line (the default)	
LINETYPE_DOT	- Dotted line	
LINETYPE_SHORTDASH	- Short-dashed line	
LINETYPE_DASHDOT	- Dash-dot line	
LINETYPE_DOUBLEDOT	- Double-dotted line	
LINETYPE_LONGDASH	- Long-dashed line	
LINETYPE_DASHDOUBLEDOT	- Dash-double-dot line	
LINETYPE_SOLID	- Solid line	
LINETYPE_ALTERNATE	- Alternate pels on	
LINETYPE_INVISIBLE	- Invisible line	

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetLineType - Remarks

A nonsolid line type consists of a sequence of "on" and "off" runs of pels that gives the appearance of a dotted or a dashed line, for example.

This attribute specifies the cosmetic line type, which is used for all line and curve drawing. It does not depend upon transforms, so that, for example, dashes do not become longer when a "zoom in" occurs.

The standard line types are implemented on each device to give a good appearance on that device, taking into account the pel resolution. Their definitions cannot be changed by applications, nor may applications define additional cosmetic line types.

The system maintains position within the line-type definition so that, for example, a curve may be implemented as a polyline. However, some functions cause position to be reset to the start of the definition. These are:

- GpiCallSegmentMatrix
- GpiMove
- GpiPop (or end of called segment) that pops current position or a model transform
- GpiSetCurrentPosition
- GpiSetLineType
- GpiSetModelTransformMatrix
- GpiSetPageViewport
- GpiSetSegmentTransformMatrix.

The default line-type is solid. This can be changed with GpiSetDefAttrs.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the line-type attribute is preserved.

GpiSetLineType - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LINE_TYPE_ATTR (0x2095)
: An invalid line type attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetLineType - Related Functions

Related Functions

- GpiBox
- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineWidth
- GpiSetLineWidthGeom

GpiSetLineType - Graphic Elements and Orders

Element Type: **OCODE_GSLT**
: This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

 Order: Set Line Type

Element Type: **OCODE_GPSLT**
: This element type is generated if the attribute mode is set to AM_PRESERVE.

 Order: Push and Set Line Type

GpiSetLineType - Example Code

This function sets the line-type to be round (as opposed to bevel or miter).

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
```

```
HPS hps;         /*   Presentation-space */
                 /*   handle.            */


GpiSetLineType(hps,
               LINETYPE_DEFAULT);
```

GpiSetLineType - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetLineWidth

GpiSetLineWidth - Syntax

This function sets the current cosmetic line-width attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;          /*  Presentation-space handle. */
FIXED    fxLineWidth;  /*  Line-width multiplier. */
BOOL     rc;           /*  Success indicator. */

rc = GpiSetLineWidth(hps, fxLineWidth);
```

GpiSetLineWidth Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetLineWidth Parameter - fxLineWidth

fxLineWidth (FIXED) - input
Line-width multiplier.

LINEWIDTH_DEFAULT
Use default; same as LINEWIDTH_NORMAL (unless changed with GpiSetDefAttrs).

LINEWIDTH_NORMAL
Normal width (1.0).

Any other positive value is a multiplier on the "normal" line width.

LINEWIDTH_THICK
Thick.

Where only two line thicknesses, "normal" and "thick", are supported, "normal" will be used for values less than or equal to 1.0 (other than LINEWIDTH_DEFAULT), and "thick" otherwise.

See "DevQueryCaps" in the *Presentation Manager Programming Reference* (CAPS_ADDITIONAL_GRAPHICS and CAPS_LINEWIDTH_THICK).

GpiSetLineWidth Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiSetLineWidth - Parameters

hps (HPS) - input
Presentation-space handle.

fxLineWidth (FIXED) - input
Line-width multiplier.

LINEWIDTH_DEFAULT
Use default; same as LINEWIDTH_NORMAL (unless changed with GpiSetDefAttrs).

LINEWIDTH_NORMAL
Normal width (1.0).

Any other positive value is a multiplier on the "normal" line width.

LINEWIDTH_THICK
Thick.

Where only two line thicknesses, "normal" and "thick", are supported, "normal" will be used for values less than or equal to 1.0 (other than LINEWIDTH_DEFAULT), and "thick" otherwise.

See "DevQueryCaps" in the *Presentation Manager Programming Reference* (CAPS_ADDITIONAL_GRAPHICS and CAPS_LINEWIDTH_THICK).

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetLineWidth - Remarks

The cosmetic line width specifies a multiplier on the "normal" line thickness for the device. Cosmetic thickness does not depend upon transforms, so that, for example, lines do not become thicker when a "zoom-in" occurs.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the line-width attribute is preserved.

GpiSetLineWidth - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LINE_WIDTH_ATTR (0x2096)
An invalid line width attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_UNSUPPORTED_ATTR_VALUE (0x210A)
An attribute value was specified with GpiSetAttrs that is not supported.

GpiSetLineWidth - Related Functions

Related Functions

- GpiBox
- GpiLine
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidthGeom

GpiSetLineWidth - Graphic Elements and Orders

Element Type: **OCODE_GSFLW**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Fractional Line Width

Element Type: **OCODE_GPSFLW**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Fractional Line Width

GpiSetLineWidth - Example Code

This function sets the line width to the default, so that there is no multiplying factor.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

GpiSetLineWidth(hps,
              LINEWIDTH_NORMAL);
```

GpiSetLineWidth - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetLineWidthGeom

GpiSetLineWidthGeom - Syntax

This function sets the current geometric line-width attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;         /*  Presentation-space handle. */
LONG    lLineWidth;  /*  Geometric line width. */
BOOL    rc;          /*  Success indicator. */

rc = GpiSetLineWidthGeom(hps, lLineWidth);
```

GpiSetLineWidthGeom Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetLineWidthGeom Parameter - lLineWidth

lLineWidth (LONG) - input
Geometric line width.

The geometric line width in world coordinates. It must not be negative.

A thickness of 0 results in an area of 0 width. Because filling includes the boundaries, this results in the thinnest possible lines and arcs, regardless of what transforms are in force.

The initial default value of the geometric line width is 1. This can be changed with GpiSetDefAttrs. Note: The default line width has been 1 since version 2.0 of the operating system. Prior to that, a wider default line width was used.

GpiSetLineWidthGeom Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetLineWidthGeom - Parameters

hps (HPS) - input
Presentation-space handle.

lLineWidth (LONG) - input
Geometric line width.

The geometric line width in world coordinates. It must not be negative.

A thickness of 0 results in an area of 0 width. Because filling includes the boundaries, this results in the thinnest possible lines and arcs, regardless of what transforms are in force.

The initial default value of the geometric line width is 1. This can be changed with GpiSetDefAttrs. Note: The default line width has been 1 since version 2.0 of the operating system. Prior to that, a wider default line width was used.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetLineWidthGeom - Remarks

The geometric line-width attribute is used only in the GpiModifyPath function (with a *lMode* of MPATH_STROKE) or in the GpiStrokePath function. This attribute specifies the width to be used in converting the lines and arcs, of which the path is composed, into wide lines and arcs. The resulting shape is treated like an area, so the boundaries are considered to be part of its interior. This means that the width of the lines and arcs is one pel wider than the geometric line width transformed to device coordinates.

The geometric line width is specified in world-coordinate units, so that, for example, the thickness varies on a zoom operation.

Normal line and curve drawing uses only the cosmetic line width (see GpiSetLineWidth).

This function must not be issued within an area or path bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the geometric line width is preserved.

GpiSetLineWidthGeom - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_GEOM_LINE_WIDTH_ATTR (0x2078)
An invalid geometric line width attribute value was specified.

GpiSetLineWidthGeom - Related Functions

Related Functions

- GpiBeginPath
- GpiCloseFigure
- GpiEndPath

- GpiFillPath
- GpiLine
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiPolyLine
- GpiQueryLineEnd
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetClipPath
- GpiSetDefAttrs
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiStrokePath

GpiSetLineWidthGeom - Graphic Elements and Orders

Element Type: **OCODE_GSSLW**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Stroke Line Width

Element Type: **OCODE_GPSSLW**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Stroke Line Width

GpiSetLineWidthGeom - Example Code

This function sets the line width geometry to double the default of 1.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

GpiSetLineWidthGeom(hps,
                    2L);
```

GpiSetLineWidthGeom - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks

Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetMarker

GpiSetMarker - Syntax

This function sets the value of the marker-symbol attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;      /*  Presentation-space handle. */
LONG    lSymbol;  /*  Marker symbol. */
BOOL    rc;       /*  Success indicator. */

rc = GpiSetMarker(hps, lSymbol);
```

GpiSetMarker Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetMarker Parameter - lSymbol

lSymbol (LONG) - input
Marker symbol.

The identity of the required marker symbol. Zero selects the default marker symbol, a value in the range 1 through 255 identifies a symbol in the current marker set. Valid values in the default marker set are shown below, these symbols are not necessarily available with other marker sets:

MARKSYM_DEFAULT	The default; same as MARKSYM_CROSS
MARKSYM_CROSS	
MARKSYM_PLUS	
MARKSYM_DIAMOND	
MARKSYM_SQUARE	
MARKSYM_SIXPOINTSTAR	
MARKSYM_EIGHTPOINTSTAR	
MARKSYM_SOLIDDIAMOND	
MARKSYM_SOLIDSQUARE	
MARKSYM_DOT	
MARKSYM_SMALLCIRCLE	
MARKSYM_BLANK	(blank)

GpiSetMarker Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetMarker - Parameters

hps (HPS) - input
Presentation-space handle.

lSymbol (LONG) - input
Marker symbol.

The identity of the required marker symbol. Zero selects the default marker symbol, a value in the range 1 through 255 identifies a symbol in the current marker set. Valid values in the default marker set are shown below, these symbols are not necessarily available with other marker sets:

MARKSYM_DEFAULT	The default; same as MARKSYM_CROSS
MARKSYM_CROSS	
MARKSYM_PLUS	
MARKSYM_DIAMOND	
MARKSYM_SQUARE	
MARKSYM_SIXPOINTSTAR	
MARKSYM_EIGHTPOINTSTAR	
MARKSYM_SOLIDDIAMOND	
MARKSYM_SOLIDSQUARE	
MARKSYM_DOT	
MARKSYM_SMALLCIRCLE	
MARKSYM_BLANK	(blank)

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetMarker - Remarks

This function must not be issued in an area bracket.

The default marker-symbol is a cross. This can be changed with GpiSetDefAttrs.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the marker attribute is to be preserved.

GpiSetMarker - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MARKER_SYMBOL_ATTR (0x209A)
An invalid marker symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetMarker - Related Functions

Related Functions

- GpiMarker
- GpiPolyMarker
- GpiQueryMarker
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetMix

GpiSetMarker - Graphic Elements and Orders

Element Type: **OCODE_GSMT**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Marker Symbol

Element Type: **OCODE_GPSMT**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Marker Symbol

GpiSetMarker - Example Code

This function changes the marker from the default (a cross) to a diamond.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

GpiSetMarker(hps,
             MARKSYM_DIAMOND);
```

GpiSetMarker - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetMarkerBox

GpiSetMarkerBox - Syntax

This function sets the current marker-box attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;         /*  Presentation-space handle. */
PSIZEF    psizfxSize;  /*  Size of marker box. */
BOOL      rc;          /*  Success indicator. */

rc = GpiSetMarkerBox(hps, psizfxSize);
```

GpiSetMarkerBox Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetMarkerBox Parameter - psizfxSize

psizfxSize (PSIZEF) - input
Size of marker box.

The size is specified in world coordinates. The fractional part of the value should be 0.

GpiSetMarkerBox Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetMarkerBox - Parameters

hps (HPS) - input
Presentation-space handle.

psizfxSize (PSIZEF) - input
Size of marker box.

The size is specified in world coordinates. The fractional part of the value should be 0.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetMarkerBox - Remarks

The value of the marker-box attribute affects the size of markers that are selected from a vector font only. The size of markers that are selected from an image font is not affected by this attribute.

For default markers, this attribute only has an effect if the device supports the scaling of default markers, that is, the CAPS_SCALED_DEFAULT_MARKERS parameter in the CAPS_ADDITIONAL_GRAPHICS element of the device capabilities array returned by the DevQueryCaps function is set to 1.

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the marker-box attribute is preserved.

The initial default value of the marker box is the size returned by DevQueryCaps (CAPS_MARKER_WIDTH and CAPS_MARKER_HEIGHT), for the currently associated device, converted to presentation page space.

The default value can be changed with GpiSetDefAttrs.

GpiSetMarkerBox - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

GpiSetMarkerBox - Related Functions

Related Functions

- GpiMarker
- GpiPolyMarker
- GpiQueryMarkerBox
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMarker
- GpiSetMarkerSet
- GpiSetMix

GpiSetMarkerBox - Graphic Elements and Orders

Element Type: **OCODE_GSMC**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Marker Cell

Element Type: **OCODE_GPSMC**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Marker Cell

GpiSetMarkerBox - Example Code

This function sets the marker box to 10x10.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

SIZEF fxSize = {MAKEFIXED(10,0),
                MAKEFIXED(10,0)};
                       /* The size is specified in */
                       /* world coordinates. The   */
                       /* fractional part of the   */
                       /* value should be zero.    */
GpiSetMarkerBox(hps,
                &fxSize);
```

GpiSetMarkerBox - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetMarkerSet

GpiSetMarkerSet - Syntax

This function sets the current marker-set attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
LONG    lSet;  /*  Marker-set local identifier. */
BOOL    rc;    /*  Success indicator. */

rc = GpiSetMarkerSet(hps, lSet);
```

GpiSetMarkerSet Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetMarkerSet Parameter - lSet

lSet (LONG) - input
Marker-set local identifier.

The identity (lcid) of the required marker set:

LCID_DEFAULT
: Default (can be set explicitly with GpiSetDefAttrs)

1-254
: Identifies a logical font.

GpiSetMarkerSet Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetMarkerSet - Parameters

hps (HPS) - input
Presentation-space handle.

lSet (LONG) - input
Marker-set local identifier.

The identity (lcid) of the required marker set:

LCID_DEFAULT
: Default (can be set explicitly with GpiSetDefAttrs)

1-254
: Identifies a logical font.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetMarkerSet - Remarks

This function must not be issued in an area bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the marker-set attribute is preserved.

If the default marker set is changed (using GpiSetDefAttrs) the initial default marker set cannot be selected with GpiSetMarkerSet.

GpiSetMarkerSet - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MARKER_SET_ATTR (0x2099)
An invalid marker set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_HUGE_FONTS_NOT_SUPPORTED (0x2035)
An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

GpiSetMarkerSet - Related Functions

Related Functions

- GpiMarker
- GpiPolyMarker
- GpiQueryMarkerSet
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetMarker
- GpiSetMarkerBox

GpiSetMarkerSet - Graphic Elements and Orders

Element Type: **OCODE_GSMS**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Marker Set

Element Type: **OCODE_GPSMS**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Marker Set

GpiSetMarkerSet - Example Code

This function changes the marker set to one defined by the logical font with id 26.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
```

```
                 /*   handle.            */

GpiSetMarkerSet(hps,
                26L);
```

GpiSetMarkerSet - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetMetaFileBits

GpiSetMetaFileBits - Syntax

This call transfers metafile data from application storage into a memory metafile.

```
#define INCL_GPIMETAFILES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HMF      hmf;       /*  Metafile-memory handle. */
LONG     lOffset;   /*  Offset, in bytes, into the metafile data from the transfer must start. */
LONG     lLength;   /*  Length of the metafile data. */
PBYTE    pbBuffer;  /*  Metafile data buffer. */
BOOL     rc;        /*  Success indicator. */

rc = GpiSetMetaFileBits(hmf, lOffset, lLength,
       pbBuffer);
```

GpiSetMetaFileBits Parameter - hmf

hmf (HMF) - input
Metafile-memory handle.

GpiSetMetaFileBits Parameter - lOffset

lOffset (LONG) - input
Offset, in bytes, into the metafile data from the transfer must start.

This is used when the metafile data is too long to fit into a single application buffer. This parameter must be greater or equal to 0.

GpiSetMetaFileBits Parameter - lLength

lLength (LONG) - input
Length of the metafile data.

It must be greater or equal to 0.

GpiSetMetaFileBits Parameter - pbBuffer

pbBuffer (PBYTE) - input
Metafile data buffer.

GpiSetMetaFileBits Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetMetaFileBits - Parameters

hmf (HMF) - input
Metafile-memory handle.

lOffset (LONG) - input
Offset, in bytes, into the metafile data from the transfer must start.

This is used when the metafile data is too long to fit into a single application buffer. This parameter must be greater or equal to 0.

lLength (LONG) - input
Length of the metafile data.

It must be greater or equal to 0.

pbBuffer (PBYTE) - input
Metafile data buffer.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetMetaFileBits - Remarks

The application must ensure that the data is in the correct format. It should not have been changed since it was created by GpiQueryMetaFileBits.

The length of the metafile is increased, if necessary, to accommodate the supplied data. If the supplied data is shorter, the metafile length is not reduced. However, in this case the metafile is still valid, if the data in it is complete and otherwise correct.

GpiSetMetaFileBits - Errors

Possible returns from WinGetLastError

PMERR_INV_HMF (0x207E)
An invalid metafile handle was specified.

PMERR_INV_METAFILE_LENGTH (0x209E)
An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.

PMERR_INV_METAFILE_OFFSET (0x209F)
An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.

PMERR_METAFILE_IN_USE (0x20D9)
An attempt has been made to access a metafile that is in use by another thread.

GpiSetMetaFileBits - Related Functions

Related Functions

- GpiCopyMetaFile
- GpiDeleteMetaFile
- GpiLoadMetaFile
- GpiPlayMetaFile
- GpiQueryMetaFileBits
- GpiQueryMetaFileLength
- GpiSaveMetaFile

GpiSetMetaFileBits - Example Code

This example shows how to copy a metafile into application storage to edit the contents and then write back to the metafile using the GpiSetMetaFileBits call.

```
#define INCL_GPIMETAFILES
#include <OS2.H>

HPS hps;         /*   Presentation-space  handle. */
HMF hmf;
PBYTE pbBuffer;
LONG cBytes;
LONG lOffset;

hmf = GpiLoadMetaFile(hps, "sample.met");

/* Allocate the buffer for the metafile data. */

cBytes = GpiQueryMetaFileLength(hmf);/* gets length of metafile */

DosAllocMem((PPVOID)pbBuffer,
            cBytes,
            PAG_READ | PAG_WRITE | PAG_COMMIT);

GpiQueryMetaFileBits(
        hmf,          /* handle of metafile              */
        lOffset,      /* offset of next byte to retrieve */
        cBytes,       /* retrieves cBytes                */
        pbBuffer);    /* buffer to receive metafile data */

/* . */
/* work with the metafile */
/* . */

/* write data back to the metafile */

GpiSetMetaFileBits(hmf, lOffset, cBytes, pbBuffer);
```

GpiSetMetaFileBits - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetMix

GpiSetMix - Syntax

This function sets the current foreground mix attribute for each individual primitive type.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;       /*  Presentation-space handle. */
LONG    lMixMode;  /*  Flag indicating the color-mixing mode. */
BOOL    rc;        /*  Success indicator. */

rc = GpiSetMix(hps, lMixMode);
```

GpiSetMix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetMix Parameter - lMixMode

lMixMode (LONG) - input
Flag indicating the color-mixing mode.

Mixing other than FM_LEAVEALONE or FM_OVERPAINT is done on the physical color index. In general, this corresponds to the color index of the logical color table if an indexed color table has been realized. In other circumstances, the color that results from such a mix cannot be predicted. Nevertheless, if FM_XOR is supported for example, drawing the same object twice with a foreground mix of FM_XOR and a background mix of BM_LEAVEALONE with no intervening drawing in other mix modes, causes the object to be erased cleanly.

The currently associated device supports any of the mixes specified as supported in DevQueryCaps (CAPS_FOREGROUND_MIX_SUPPORT). Any other valid mixes may be supported for some primitive types, but otherwise results in FM_OVERPAINT. An error is raised only if the value specified is not one of those listed below.

Note: Mixes marked with an asterisk (*) are mandatory for all devices, except that FM_OR is only mandatory for devices capable of supporting it. FM_XOR is mandatory only on displays.

The possible values for this parameter are:

FM_DEFAULT
: Use default, the same as FM_OVERPAINT, unless changed with GpiSetDefAttrs

FM_OR
: Logical-OR (*)

FM_OVERPAINT
: Overpaint (*)

FM_XOR
: Logical-XOR (*)

FM_LEAVEALONE
: Leave alone (invisible) (*)

FM_AND
: Logical-AND

FM_SUBTRACT
: (Inverse source) AND destination

FM_MASKSRCNOT
: Source AND (inverse destination)

FM_ZERO
: All zeros

FM_NOTMERGESRC

Inverse (source OR destination)
FM_NOTXORSRC
Inverse (source XOR destination)
FM_INVERT
Inverse (destination)
FM_MERGESRCNOT
Source OR (inverse destination)
FM_NOTCOPYSRC
Inverse (source)
FM_MERGENOTSRC
(Inverse source) OR destination
FM_NOTMASKSRC
Inverse (source AND destination)
FM_ONE
All ones.

GpiSetMix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetMix - Parameters

hps (HPS) - input
Presentation-space handle.

lMixMode (LONG) - input
Flag indicating the color-mixing mode.

Mixing other than FM_LEAVEALONE or FM_OVERPAINT is done on the physical color index. In general, this corresponds to the color index of the logical color table if an indexed color table has been realized. In other circumstances, the color that results from such a mix cannot be predicted. Nevertheless, if FM_XOR is supported for example, drawing the same object twice with a foreground mix of FM_XOR and a background mix of BM_LEAVEALONE with no intervening drawing in other mix modes, causes the object to be erased cleanly.

The currently associated device supports any of the mixes specified as supported in DevQueryCaps (CAPS_FOREGROUND_MIX_SUPPORT). Any other valid mixes may be supported for some primitive types, but otherwise results in FM_OVERPAINT. An error is raised only if the value specified is not one of those listed below.

Note: Mixes marked with an asterisk (*) are mandatory for all devices, except that FM_OR is only mandatory for devices capable of supporting it. FM_XOR is mandatory only on displays.

The possible values for this parameter are:

FM_DEFAULT
Use default, the same as FM_OVERPAINT, unless changed with GpiSetDefAttrs
FM_OR
Logical-OR (*)
FM_OVERPAINT
Overpaint (*)
FM_XOR
Logical-XOR (*)
FM_LEAVEALONE

Leave alone (invisible) (*)

FM_AND
Logical-AND

FM_SUBTRACT
(Inverse source) AND destination

FM_MASKSRCNOT
Source AND (inverse destination)

FM_ZERO
All zeros

FM_NOTMERGESRC
Inverse (source OR destination)

FM_NOTXORSRC
Inverse (source XOR destination)

FM_INVERT
Inverse (destination)

FM_MERGESRCNOT
Source OR (inverse destination)

FM_NOTCOPYSRC
Inverse (source)

FM_MERGENOTSRC
(Inverse source) OR destination

FM_NOTMASKSRC
Inverse (source AND destination)

FM_ONE
All ones.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiSetMix - Remarks

The current values for each primitive type are updated. The attribute mode (see GpiSetAttrMode) determines whether the current value of the mix attribute is preserved.

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetMix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MIX_ATTR (0x20A3)
An invalid mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetMix - Related Functions

Related Functions

- GpiBeginArea
- GpiBox
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiEndArea
- GpiFullArc
- GpiLine
- GpiMarker
- GpiMove
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolyLine
- GpiPolyMarker
- GpiPolySpline
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryMix
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs

GpiSetMix - Graphic Elements and Orders

Element Type: **OCODE_GSMX**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Mix

Element Type: **OCODE_GPSMX**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Mix

GpiSetMix - Example Code

This function sets the current foreground mix attribute for each individual primitive type.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */
```

```
GpiSetMix(hps,
          FM_LEAVEALONE);
```

GpiSetMix - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetModelTransformMatrix

GpiSetModelTransformMatrix - Syntax

This function sets the model transform matrix for subsequent primitives.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
LONG         lCount;       /*  Number of elements in matrix. */
PMATRIXLF    pmatlfArray;  /*  Transformation matrix. */
LONG         lOptions;     /*  Transform options. */
BOOL         rc;           /*  Success indicator. */

rc = GpiSetModelTransformMatrix(hps, lCount,
       pmatlfArray, lOptions);
```

GpiSetModelTransformMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetModelTransformMatrix Parameter - lCount

lCount (LONG) - input
Number of elements in matrix.

The number of elements of *pmatlfArray* to be examined, starting from the beginning of the structure. If *lCount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. If *lCount* = 0, the identity matrix is used.

GpiSetModelTransformMatrix Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

GpiSetModelTransformMatrix Parameter - lOptions

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the *pmatlfArray* should be used to modify the existing current model transform (the existing transform is the concatenation, in the current call context, of the instance, segment and model transforms, from the root segment downwards). Possible values are:

TRANSFORM_REPLACE
The previous model transform is discarded and replaced by the specified transform.

TRANSFORM_ADD
The specified transform is combined with the existing model transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms.

TRANSFORM_PREEMPT
The specified transform is combined with the existing model transform, in the order (1) new transform, (2) existing transform.

GpiSetModelTransformMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetModelTransformMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of elements in matrix.

The number of elements of *pmatlfArray* to be examined, starting from the beginning of the structure. If *lCount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. If *lCount* = 0, the identity matrix is used.

pmatlfArray (PMATRIXLF) - input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the *pmatlfArray* should be used to modify the existing current model transform (the existing transform is the concatenation, in the current call context, of the instance, segment and model transforms, from the root segment downwards). Possible values are:

TRANSFORM_REPLACE
The previous model transform is discarded and replaced by the specified transform.

TRANSFORM_ADD
The specified transform is combined with the existing model transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms.

TRANSFORM_PREEMPT
The specified transform is combined with the existing model transform, in the order (1) new transform, (2) existing transform.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetModelTransformMatrix - Remarks

The matrix is used to update the previous current model transform, depending upon the value of *lOptions* .

The transform is specified as a one-dimensional array of *lCount* elements, being the first elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

```
Matrix                  Array

a  b  0
c  d  0           (a,b,0,c,d,0,e,f,1)
e  f  1
```

The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point:

```
(a*x + c*y + e, b*x + d*y + f)
```

If scaling values greater than unity are given (which only applies if the presentation space coordinate format as set by the GpiCreatePS function is GPIF_LONG) it is possible for the combined effect of this, and any other relevant transforms, to exceed fixed-point implementation limits. This causes an error.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the model transform is preserved.

Model transforms can apply to primitives either inside or outside segments.

GpiSetModelTransformMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
: An invalid length or count parameter was specified.

PMERR_INV_MATRIX_ELEMENT (0x209B)
: An invalid transformation matrix element was specified.

PMERR_INV_TRANSFORM_TYPE (0x20D0)
: An invalid options parameter was specified with a transform matrix function.

GpiSetModelTransformMatrix - Related Functions

Related Functions

- GpiCallSegmentMatrix
- GpiQueryModelTransformMatrix
- GpiQuerySegmentTransformMatrix
- GpiSetSegmentTransformMatrix

GpiSetModelTransformMatrix - Graphic Elements and Orders

Element Type: **OCODE_GSTM**
: This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

 Order: Set Model Transform

Element Type: **OCODE_GPSTM**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Model Transform

GpiSetModelTransformMatrix - Example Code

This function sets the model transformation matrix as one which scales everything by a factor of 2.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */
MATRIXLF matlf = { MAKEFIXED(2,0),  /* see pmgpi.h for a */
                                    /* definition of the */
                                    /* MAKEFIXED macro.  */
                   0, 0, 0,
                   MAKEFIXED(2,0),
                   0, 0, 0, 1};


GpiSetModelTransformMatrix(hps,
                           1L,
                           &matlf,
                           TRANSFORM_REPLACE);
```

GpiSetModelTransformMatrix - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetPageViewport

GpiSetPageViewport - Syntax

This function sets the page viewport within device space.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;           /*  Presentation-space handle. */
PRECTL    prclViewport;  /*  Page viewport. */
BOOL      rc;            /*  Success indicator. */

rc = GpiSetPageViewport(hps, prclViewport);
```

GpiSetPageViewport Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetPageViewport Parameter - prclViewport

prclViewport (PRECTL) - input
Page viewport.

The page viewport is specified in device units.

GpiSetPageViewport Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPageViewport - Parameters

hps (HPS) - input
Presentation-space handle.

prclViewport (PRECTL) - input
Page viewport.

The page viewport is specified in device units.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPageViewport - Remarks

The presentation page maps to the page viewport and together they define the device transform.

When a presentation space is associated with a device context, a default page viewport is set up.

The origin in device space is mapped to the bottom-left of the output media (window or paper, for example).

This function must not be issued when there is no device context associated with the presentation space.

This function is ignored if issued to a presentation space that is associated with a device context of type OD_QUEUED (with PM_Q_STD data), OD_METAFILE, orOD_METAFILE_NOQUERY.

GpiSetPageViewport - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PAGE_VIEWPORT (0x20AD)
An invalid viewport parameter was specified with GpiSetPageViewport.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetPageViewport - Related Functions

Related Functions

- GpiCreatePS
- GpiQueryPageViewport

GpiSetPageViewport - Example Code

This example sets the area of the device in which the picture is displayed to page viewport within device space.

```
#define INCL_GPITRANSFORMS
```

```
#include <OS2.H>
HPS hps;         /*   Presentation-space */
                 /*   handle.            */
RECTL rclField = {25L, /* x coordinate of left-hand edge of */
                       /* rectangle. */
                  25L, /* y coordinate of bottom edge of    */
                       /* rectangle. */
                  425L, /* x coordinate of right-hand edge of */
                       /* rectangle. */
                 425L}; /* y coordinate of top edge of
                        /* rectangle. */


GpiSetPageViewport(hps, &rclField);
```

GpiSetPageViewport - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetPaletteEntries

GpiSetPaletteEntries - Syntax

This function changes the entries in a palette.

```
#define INCL_GPILOGCOLORTABLE /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPAL      hpal;      /*  Palette handle. */
ULONG     ulFormat;  /*  Format of entries in the table. */
ULONG     ulStart;   /*  Starting index. */
ULONG     ulCount;   /*  Count of elements in aulTable. */
PULONG    aulTable;  /*  Start of the application data area. */
BOOL      rc;        /*  Success indicator. */

rc = GpiSetPaletteEntries(hpal, ulFormat,
       ulStart, ulCount, aulTable);
```

GpiSetPaletteEntries Parameter - hpal

hpal (HPAL) - input
Palette handle.

GpiSetPaletteEntries Parameter - ulFormat

ulFormat (ULONG) - input
Format of entries in the table.

LCOLF_CONSECRGB
Array of RGB values, corresponding to color indexes *ulStart* upwards. Each entry is 4 bytes long.

This is currently the only valid value for this parameter.

GpiSetPaletteEntries Parameter - ulStart

ulStart (ULONG) - input
Starting index.

GpiSetPaletteEntries Parameter - ulCount

ulCount (ULONG) - input
Count of elements in *aulTable* .

This must be greater than or equal to 0.

GpiSetPaletteEntries Parameter - aulTable

aulTable (PULONG) - input
Start of the application data area.

This contains the palette definition data. The format depends on the value of *ulFormat* .

Each color value is a 4-byte integer, with a value of

```
(F * 16777216) + (R * 65536) + (G * 256) + B
```

where:

F is a flag byte, which can take the following values (these can be ORed together if required):

	PC_RESERVED	This index is an animating index. This means that the application might frequently change the RGB value, so the system should not map the logical index of the palette of another application to the entry in the physical palette used for this color.
	PC_EXPLICIT	The low-order word of the logical color table entry designates a physical palette entry. This allows an application to show the contents of the device palette as realized for other logical palettes. This does not prevent the color in the entry from being changed for any reason.
R	is red intensity value	
G	is green intensity value	
B	is blue intensity value. The maximum intensity for each primary is 255.	

GpiSetPaletteEntries Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiSetPaletteEntries - Parameters

hpal (HPAL) - input
Palette handle.

ulFormat (ULONG) - input
Format of entries in the table.

LCOLF_CONSECRGB
Array of RGB values, corresponding to color indexes *ulStart* upwards. Each entry is 4 bytes long.

This is currently the only valid value for this parameter.

ulStart (ULONG) - input
Starting index.

ulCount (ULONG) - input
Count of elements in *aulTable* .

This must be greater than or equal to 0.

aulTable (PULONG) - input
Start of the application data area.

This contains the palette definition data. The format depends on the value of *ulFormat* .

Each color value is a 4-byte integer, with a value of

```
(F * 16777216) + (R * 65536) + (G * 256) + B
```

where:

F — is a flag byte, which can take the following values (these can be ORed together if required):

PC_RESERVED	This index is an animating index. This means that the application might frequently change the RGB value, so the system should not map the logical index of the palette of another application to the entry in the physical palette used for this color.
PC_EXPLICIT	The low-order word of the logical color table entry designates a physical palette entry. This allows an application to show the contents of the device palette as realized for other logical palettes. This does not prevent the color in the entry from being changed for any reason.

R — is red intensity value

G — is green intensity value

B — is blue intensity value. The maximum intensity for each primary is 255.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiSetPaletteEntries - Remarks

The changes made by this function do not become apparent until WinRealizePalette is called, even for animating indices. Changes can be made more rapidly using GpiAnimatePalette with animating indices, assuming that the hardware being used supports this.

GpiSetPaletteEntries can be called at any time to change a logical palette, and the physical palette of the device will incorporate the changes as best it can; however, the system cannot guarantee that a change will be realized in the hardware palette, since realization depends on whether the associated window is in the foreground and on the number of available hardware palette entries.

All presentation spaces that have this palette selected into them (see GpiSelectPalette), are updated with the effects of this function.

If a palette is selected into a presentation space that is associated with a device context of type OD_METAFILE or OD_METAFILE_NOQUERY only the final color values are recorded in the metafile. This means that, while metafiling, this function must only be used for incremental additions to the color table.

It is an error if a palette is selected into a presentation space that is within an area or path definition when this function is issued.

GpiSetPaletteEntries - Errors

Possible returns from WinGetLastError

PMERR_INV_HPAL (0x2111)
An invalid color palette handle was specified.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_COLOR_DATA (0x2054)
Invalid color table definition data was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_FORMAT (0x2055)
An invalid format parameter was specified with GpiCreateLogColorTable.

PMERR_INV_COLOR_START_INDEX (0x2058)

An invalid starting index parameter was specified with a logical color table or color query function.

PMERR_INSUFFICIENT_MEMORY (0x203E)
The operation terminated through insufficient memory.

PMERR_PALETTE_BUSY (0x2112)
An attempt has been made to reset the owner of a palette when it was busy.

PMERR_INV_IN_AREA (0x2085)
An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.

GpiSetPaletteEntries - Related Functions

Related Functions

- GpiAnimatePalette
- GpiCreatePalette
- GpiDeletePalette
- GpiQueryPalette
- GpiQueryPaletteInfo
- GpiSelectPalette

GpiSetPaletteEntries - Example Code

This example changes the entries in a palette.

```
#define INCL_GPILOGCOLORTABLE
#include <OS2.H>

HPAL hpal;  /* palette handle */
UINT R, G, B;
typedef struct ENTRY
{
ULONG index;
ULONG pal_def;
}Entry;

struct TABLE
{
Entry entry1;
Entry entry2;
Entry entry3;
}Table;
BYTE F = PC_RESERVED;

/* In our table, there are 3 8-byte entries. The first 4 bytes */
/* of each entry represent the index and the second 4 bytes of */
/* each entry represent the value of the following formula:    */
/*                                                             */
/*    (F * 16777216) + (R * 65536) + (G * 256) + B             */
/*                                                             */
/* which is the palette definition.                            */
/* where F is the flag PC_RESERVED and R,G,B are the red,      */
/* green, and blue intensity values respectively.              */

F = 10; R = 10; G = 10;
Table.entry1.pal_def = (F * 16777216)+(R * 65536)+(G * 256) + B;
Table.entry1.index = 0L;

F = 25; R = 25; G = 25;
Table.entry2.pal_def = (F * 16777216)+(R * 65536)+(G * 256) + B;
Table.entry2.index = 1L;
```

```
F = 40; R = 40; G = 40;
Table.entry3.pal_def = (F * 16777216)+(R * 65536)+(G * 256) + B;
Table.entry3.index = 2L;

GpiSetPaletteEntries(hpal,
                     LCOLF_CONSECRGB, /* Array of RGB values,   */
                                      /* corresponding to color */
                                      /* indexes lStart         */
                                      /* upwards. Each entry    */
                                      /* is 4 bytes long.       */
                     0L,              /* start at zero.         */
                     3L,              /* elements in table.     */
       &Table.entry1.index);          /* first element in table. */
```

GpiSetPaletteEntries - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetPattern

GpiSetPattern - Syntax

This function sets the current value of the pattern-symbol attribute.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS     hps;             /*  Presentation-space handle. */
LONG    lPatternSymbol;  /*  Pattern symbol. */
BOOL    rc;              /*  Success indicator. */

rc = GpiSetPattern(hps, lPatternSymbol);
```

GpiSetPattern Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetPattern Parameter - lPatternSymbol

lPatternSymbol (LONG) - input
Pattern symbol.

Identifies the shading pattern to be used to fill areas. The pattern that appears depends on the particular pattern set selected by the pattern-set attribute. A value of 0 selects the default pattern and values in the range 1 through 255 select particular patterns within the set.

Possible values if the default pattern set has been selected are shown in the following table:

Symbolic name	Description	Pattern number (see Figure under Notes)
PATSYM_DEFAULT	The default; same as PATSYM_SOLID (unless changed with GpiSetDefAttrs).	
PATSYM_DENSE1 through PATSYM_DENSE8	Solid shading with decreasing density	1 through 8
PATSYM_VERT	Vertical pattern	9
PATSYM_HORIZ	Horizontal pattern	10
PATSYM_DIAG1	Diagonal pattern 1, bottom left to top right	11
PATSYM_DIAG2	Diagonal pattern 2, bottom left to top right	12
PATSYM_DIAG3	Diagonal pattern 3, top left to bottom right	13
PATSYM_DIAG4	Diagonal pattern 4, top left to bottom right	14
PATSYM_NOSHADE	No shading	15
PATSYM_SOLID	Solid shading	16
PATSYM_HALFTONE	Alternate pels set on	
PATSYM_BLANK	Blank (same as PATSYM_NOSHADE)	

Note: The pattern PATSYM_HALFTONE can be the same as PATSYM_DENSE4. On non bit-mapped devices it may be mapped to another base pattern.

If the specified pattern is not valid, the default (device-dependent) pattern is used.

GpiSetPattern Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPattern - Parameters

hps (HPS) - input
Presentation-space handle.

lPatternSymbol (LONG) - input
Pattern symbol.

Identifies the shading pattern to be used to fill areas. The pattern that appears depends on the particular pattern set selected by the pattern-set attribute. A value of 0 selects the default pattern and values in the range 1 through 255 select particular patterns within the set.

Possible values if the default pattern set has been selected are shown in the following table:

Symbolic name	Description	Pattern number (see Figure under Notes)
PATSYM_DEFAULT	The default; same as PATSYM_SOLID (unless changed with GpiSetDefAttrs).	
PATSYM_DENSE1 through PATSYM_DENSE8	Solid shading with decreasing density	1 through 8
PATSYM_VERT	Vertical pattern	9
PATSYM_HORIZ	Horizontal pattern	10
PATSYM_DIAG1	Diagonal pattern 1, bottom left to top right	11
PATSYM_DIAG2	Diagonal pattern 2, bottom left to top right	12
PATSYM_DIAG3	Diagonal pattern 3, top left to bottom right	13
PATSYM_DIAG4	Diagonal pattern 4, top left to bottom right	14
PATSYM_NOSHADE	No shading	15

```
PATSYM_SOLID     Solid shading    16

PATSYM_HALFTONE  Alternate pels
                 set on

PATSYM_BLANK     Blank (same as
                 PATSYM_NOSHADE)
```

Note: The pattern PATSYM_HALFTONE can be the same as PATSYM_DENSE4. On non bit-mapped devices it may be mapped to another base pattern.

If the specified pattern is not valid, the default (device-dependent) pattern is used.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion

FALSE
Error occurred.

GpiSetPattern - Remarks

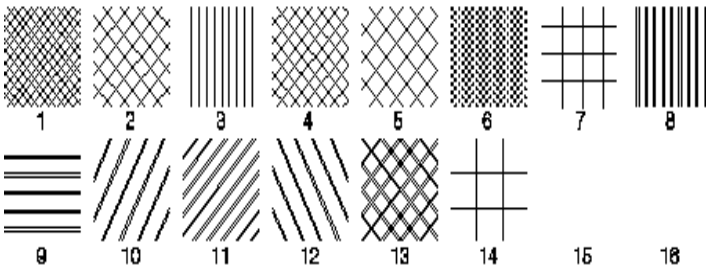
Any symbol from a raster font can be used as a pattern by the appropriate use of this function and the GpiSetPatternSet function.

If the current pattern set specifies a bit map (see GpiSetBitmapId and GpiSetPatternSet), the pattern attribute is ignored.

If *lPatternSymbol* is set or defaulted to PATSYM_SOLID, and the *lSet* parameter of GpiSetPatternSet is LCID_DEFAULT, pattern colors that are not available may be approximated by dithering (unless dithering has been disabled by setting the LCOL_PURECOLOR bit on the *flOptions* parameter of GpiCreateLogColorTable).

This function must not be issued in an area or path bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the pattern symbol is preserved.



GpiSetPattern - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATTERN_ATTR (0x20B0)
: An invalid pattern symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

GpiSetPattern - Related Functions

Related Functions

- GpiBeginArea
- GpiEndArea
- GpiQueryPattern
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix
- GpiSetPatternRefPoint
- GpiSetPatternSet

GpiSetPattern - Graphic Elements and Orders

Element Type: **OCODE_GSPT**
: This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

 Order: Set Pattern Symbol

Element Type: **OCODE_GPSPT**
: This element type is generated if the attribute mode is set to AM_PRESERVE.

 Order: Push and Set Pattern Symbol

GpiSetPattern - Example Code

This function sets the current value of the pattern-symbol to horizontal. This means that when areas are filled, they are filled with a horizontal

shading pattern.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
HPS hps;          /*   Presentation-space */
                  /*   handle.            */

GpiSetPattern(hps,PATSYM_HORIZ);
```

GpiSetPattern - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetPatternRefPoint

GpiSetPatternRefPoint - Syntax

This function sets the current pattern reference point to the specified value.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;           /*  Presentation-space handle. */
PPOINTL    pptlRefPoint;  /*  Pattern reference point. */
BOOL       rc;            /*  Success indicator. */

rc = GpiSetPatternRefPoint(hps, pptlRefPoint);
```

GpiSetPatternRefPoint Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetPatternRefPoint Parameter - pptlRefPoint

pptlRefPoint (PPOINTL) - input
Pattern reference point.

The coordinates are world coordinates.

GpiSetPatternRefPoint Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPatternRefPoint - Parameters

hps (HPS) - input
Presentation-space handle.

pptlRefPoint (PPOINTL) - input
Pattern reference point.

The coordinates are world coordinates.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPatternRefPoint - Remarks

The pattern reference point is the point to which the origin of the area filling pattern maps. The pattern is mapped into the area to be filled by conceptually replicating the pattern definition in a horizontal and vertical direction.

Because the pattern reference point is subject to all of the transforms, if an area is moved by changing a transform and redrawing, the fill pattern also appears to move, so as to retain its position relative to the area boundaries.

The pattern reference point, which is specified in world coordinates, need not be inside the actual area to be filled. The pattern reference point is not subject to clipping.

This function must not be issued in an area or path bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the pattern reference point is preserved.

The initial default pattern reference point is (0,0). This can be changed with GpiSetDefAttrs.

GpiSetPatternRefPoint - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetPatternRefPoint - Related Functions

Related Functions

- GpiBeginArea
- GpiEndArea
- GpiQueryPatternRefPoint
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetPattern
- GpiSetPatternSet

GpiSetPatternRefPoint - Graphic Elements and Orders

Element Type: **OCODE_GSPRP**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Pattern Reference Point

Element Type: **OCODE_GPSPRP**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Pattern Reference Point

GpiSetPatternRefPoint - Example Code

This function sets the current pattern reference point to the specified value.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
HPS hps;          /*   Presentation-space */
                  /*   handle.            */
POINTL ptlRefPoint = {0,0};

GpiSetPatternRefPoint(hps, &ptlRefPoint);
```

GpiSetPatternRefPoint - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetPatternSet

GpiSetPatternSet - Syntax

This function sets the current pattern-set attribute to the specified value.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
LONG    lSet;  /*  Pattern-set local identifier. */
BOOL    rc;    /*  Success indicator. */

rc = GpiSetPatternSet(hps, lSet);
```

GpiSetPatternSet Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetPatternSet Parameter - lSet

lSet (LONG) - input
Pattern-set local identifier.

LCID_DEFAULT
Default (can be set explicitly with GpiSetDefAttrs).
1-254
Identifies a logical font or a bit map.

GpiSetPatternSet Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPatternSet - Parameters

hps (HPS) - input
Presentation-space handle.

lSet (LONG) - input
Pattern-set local identifier.

LCID_DEFAULT
Default (can be set explicitly with GpiSetDefAttrs).
1-254
Identifies a logical font or a bit map.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPatternSet - Remarks

The bit map, or character within the font selected, is used for shading. On some devices, a simplified form of the bit map, or character, is used. For example, only a subset such as the first 8 by 8 pels may be used; also on a monochrome device a color bit map is converted to monochrome.

Some fonts are not suitable, and an error is returned if an attempt is made to set them as the current pattern set. These include device fonts that cannot be used for shading, and any kind of raster font for a plotter device.

This function must not be issued in an area or path bracket.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the pattern-set attribute is preserved.

If the default pattern set is changed (using GpiSetDefAttrs), the initial default pattern marker set cannot be selected with GpiSetPatternSet.

GpiSetPatternSet - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATTERN_SET_ATTR (0x20B2)
: An invalid pattern set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.

PMERR_INV_PATTERN_SET_FONT (0x20B3)
: An attempt was made to use an unsuitable font as a pattern set.

PMERR_HUGE_FONTS_NOT_SUPPORTED (0x2035)
: An attempt was made using GpiSetCharSet, GpiSetPatternSet, GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.

GpiSetPatternSet - Related Functions

Related Functions

- GpiBeginArea
- GpiCreateLogFont
- GpiEndArea
- GpiQueryPatternSet
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetPattern
- GpiSetPatternRefPoint

GpiSetPatternSet - Graphic Elements and Orders

Element Type: **OCODE_GSPS**
: This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

 Order: Set Pattern Set

Element Type: **OCODE_GPSPS**
: This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Pattern Set

GpiSetPatternSet - Example Code

This function sets the current pattern-set attribute to the logical font with id 35.

```
#define INCL_GPIPRIMITIVES
#include <OS2.H>
HPS hps;          /*   Presentation-space */
                  /*   handle.            */


GpiSetPatternSet(hps, 35L);
```

GpiSetPatternSet - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetPel

GpiSetPel - Syntax

This function sets a pel, at a position specified in world coordinates, using the current (line) color and mix.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;        /*  Presentation-space handle. */
PPOINTL    pptlPoint;  /*  Position in world coordinates. */
LONG       lHits;      /*  Correlation and error indicators. */

lHits = GpiSetPel(hps, pptlPoint);
```

GpiSetPel Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetPel Parameter - pptlPoint

pptlPoint (PPOINTL) - input
Position in world coordinates.

GpiSetPel Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiSetPel - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPoint (PPOINTL) - input
Position in world coordinates.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiSetPel - Remarks

This function is subject to all the usual clipping (clip path, clip region, viewing limits, graphics field, visible region), and no error is returned if the point is subject to clipping.

This function is independent of drawing mode (see GpiSetDrawingMode); the effect always occurs immediately, and it is not retained even if the drawing mode is **draw-and-retain** or **retain**. (Its effect is, however, recorded in a metafile, but note that this is only successful if the metafile is replayed on a similar device, with **draw** drawing mode.)

Note: This function must not be used when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiSetPel - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetPel - Related Functions

Related Functions

- GpiQueryPel
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetColor
- GpiSetDefAttrs
- GpiSetMix

GpiSetPel - Example Code

This function sets a pel, at a position specified in world coordinates, using the current (line) color and mix.

```
#define INCL_GPIBITMAPS
#include <OS2.H>
HPS hps;         /*   Presentation-space */
                 /*   handle.            */


POINTL ptlPoint = {0,0};


GpiSetPel(hps, &ptlPoint);
```

GpiSetPel - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetPickAperturePosition

GpiSetPickAperturePosition - Syntax

This function sets the center of the pick aperture, in presentation page space, for subsequent nonretained correlation operations.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS        hps;       /*  Presentation-space handle. */
PPOINTL    pptlPick;  /*  Center of the pick aperture. */
BOOL       rc;        /*  Success indicator. */

rc = GpiSetPickAperturePosition(hps, pptlPick);
```

GpiSetPickAperturePosition Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetPickAperturePosition Parameter - pptlPick

pptlPick (PPOINTL) - input
Center of the pick aperture.

The center is in presentation page coordinates.

GpiSetPickAperturePosition Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPickAperturePosition - Parameters

hps (HPS) - input
Presentation-space handle.

pptlPick (PPOINTL) - input
Center of the pick aperture.

The center is in presentation page coordinates.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPickAperturePosition - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetPickAperturePosition - Related Functions

Related Functions

- GpiQueryPickAperturePosition
- GpiQueryPickApertureSize
- GpiSetPickApertureSize

GpiSetPickAperturePosition - Example Code

In this example we query the position of the center of the pick aperture.

```
#define INCL_GPICORELATION
#include <OS2.H>

BOOL flResult;
HPS hps;              /* Presentation space handle. */
POINTL ptlPoint = {50L, 50L};  /* Pick-aperture position. */

flResult = GpiSetPickAperturePosition(hps, &ptlPoint);
```

GpiSetPickAperturePosition - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Example Code
Related Functions
Glossary

GpiSetPickApertureSize

GpiSetPickApertureSize - Syntax

This function sets the pick-aperture size.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
LONG      lOptions;   /*  Setting option. */
PSIZEL    psizlSize;  /*  Pick aperture size. */
BOOL      rc;         /*  Success indicator. */

rc = GpiSetPickApertureSize(hps, lOptions,
       psizlSize);
```

GpiSetPickApertureSize Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetPickApertureSize Parameter - lOptions

lOptions (LONG) - input
Setting option.

PICKAP_DEFAULT
Use the default pick aperture. The value of *psizlSize* is ignored.
PICKAP_REC
Use the values specified by *psizlSize* .

GpiSetPickApertureSize Parameter - psizlSize

psizlSize (PSIZEL) - input
Pick aperture size.

GpiSetPickApertureSize Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPickApertureSize - Parameters

hps (HPS) - input
Presentation-space handle.

lOptions (LONG) - input
Setting option.

PICKAP_DEFAULT
: Use the default pick aperture. The value of *psizlSize* is ignored.

PICKAP_REC
: Use the values specified by *psizlSize* .

psizlSize (PSIZEL) - input
: Pick aperture size.

rc (BOOL) - returns
: Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetPickApertureSize - Remarks

The pick aperture can be set either to the default value, or to a specified size in presentation page space. This is used in any subsequent nonretained or retained correlation operations.

The default size is a rectangle in presentation page space that produces a square on the device, with side equal to the default character cell height.

GpiSetPickApertureSize - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PICK_APERTURE_OPTION (0x20B4)
: An invalid options parameter was specified with GpiSetPickApertureSize.

PMERR_INV_PICK_APERTURE_SIZE (0x20B6)
: An invalid size parameter was specified with GpiSetPickApertureSize.

GpiSetPickApertureSize - Related Functions

Related Functions

- GpiQueryPickApertureSize
- GpiQueryPickAperturePosition
- GpiSetPickAperturePosition

GpiSetPickApertureSize - Example Code

In this example we set the pick-aperture size to a 4 by 4 box in world coordinates.

```
#define INCL_GPICORRELATION
#include <OS2.H>

HPS hps;                /* Presentation space handle. */
SIZEL sizel;            /* Pick-aperture position.    */

sizel.cx = 4L;  sizel.cy = 4L;
GpiSetQueryPickApertureSize(hps, &sizel);
```

GpiSetPickApertureSize - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetPS

GpiSetPS - Syntax

This function sets the presentation space size, units, and format.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;        /*  Presentation-space handle. */
PSIZEL    psizlsize;  /*  Presentation-space size. */
ULONG     flOptions;  /*  Options. */
BOOL      rc;         /*  Success indicator. */

rc = GpiSetPS(hps, psizlsize, flOptions);
```

GpiSetPS Parameter - hps

hps (HPS) - input

Presentation-space handle.

GpiSetPS Parameter - psizlsize

psizlsize (PSIZEL) - input
Presentation-space size.

GpiSetPS Parameter - flOptions

flOptions (ULONG) - input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

PS_UNITS
Presentation page size units.

Indicates the units for the presentation page size. In each case, the origin is at the bottom left. Possible values are shown in the following list:

PU_ARBITRARY
Application-convenient units
PU_PELS
Pel coordinates
PU_LOMETRIC
Units of 0.1 mm
PU_HIMETRIC
Units of 0.01 mm
PU_LOENGLISH
Units of 0.01 inch
PU_HIENGLISH
Units of 0.001 inch
PU_TWIPS
Units of 1/1440 inch.

PS_FORMAT
Coordinate format.

Indicates options to be used when storing coordinate values internally in the segment store.

For most calls, the format is not directly visible to an application. However, it is visible during editing (for example, GpiQueryElement). The format also has an effect on the amount of storage required for segment store.

One of these can be selected, for a GPIT_NORMAL presentation space (for a GPIT_MICRO presentation space, only GPIF_DEFAULT is allowed):

GPIF_DEFAULT
Default local format (same as GPIF_LONG)
GPIF_SHORT
2-byte integers
GPIF_LONG
4-byte integers.

PS_NORESET
Inhibit full reset indicator.

Inhibits the full reset of the presentation space. If this flag is set, a reset equivalent to GRES_SEGMENTS is

performed. If it is not set, a full reset (GRES_ALL) is performed.

GpiSetPS Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPS - Parameters

hps (HPS) - input
Presentation-space handle.

psizlsize (PSIZEL) - input
Presentation-space size.

flOptions (ULONG) - input
Options.

This contains fields of option bits. For each field, one value should be selected (unless the default is suitable). These values can then be ORed together to generate the parameter.

PS_UNITS
Presentation page size units.

Indicates the units for the presentation page size. In each case, the origin is at the bottom left. Possible values are shown in the following list:

PU_ARBITRARY
Application-convenient units
PU_PELS
Pel coordinates
PU_LOMETRIC
Units of 0.1 mm
PU_HIMETRIC
Units of 0.01 mm
PU_LOENGLISH
Units of 0.01 inch
PU_HIENGLISH
Units of 0.001 inch
PU_TWIPS
Units of 1/1440 inch.

PS_FORMAT
Coordinate format.

Indicates options to be used when storing coordinate values internally in the segment store.

For most calls, the format is not directly visible to an application. However, it is visible during editing (for example, GpiQueryElement). The format also has an effect on the amount of storage required for segment store.

One of these can be selected, for a GPIT_NORMAL presentation space (for a GPIT_MICRO presentation space, only GPIF_DEFAULT is allowed):

GPIF_DEFAULT
Default local format (same as GPIF_LONG)
GPIF_SHORT
2-byte integers
GPIF_LONG
4-byte integers.

PS_NORESET
Inhibit full reset indicator.

Inhibits the full reset of the presentation space. If this flag is set, a reset equivalent to GRES_SEGMENTS is performed. If it is not set, a full reset (GRES_ALL) is performed.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetPS - Remarks

The presentation space is re-initialized to the same state that occurs as if it had been created using the specified size and option values; however, whether the presentation space is a micro presentation space or a normal presentation space cannot be changed, and any device context that is already associated remains associated.

The presentation space code page is set to the current process code page.

On completion, the presentation space is reset with the equivalent of GRES_ALL (see GpiResetPS), unless *PS_NORESET* is specified, in which case only the equivalent of a GRES_SEGMENTS reset is performed.

This function cannot be used to a presentation space that is associated with a device context of type OD_QUEUED, OD_METAFILE, or OD_METAFILE_NOQUERY.

The following options for *flOptions* are ignored:

PS_TYPE	Presentation space.
PS_MODE	Mode
PS_ASSOCIATE	Association indicator

GpiSetPS - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_HDC (0x207C)
An invalid device-context handle or (micro presentation space) presentation-space handle was specified.

PMERR_INV_PS_SIZE (0x20BA)
An invalid size parameter was specified with GpiCreatePS or GpiSetPS.

PMERR_INV_OR_INCOMPAT_OPTIONS (0x20A9)
An invalid or incompatible (with micro presentation space) options parameter was specified with GpiCreatePS or GpiSetPS.

PMERR_INV_FOR_THIS_DC_TYPE (0x2074)
An attempt has been made to issue GpiRemoveDynamics or GpiDrawDynamics to a presentation space associated with a metafile device context.

GpiSetPS - Related Functions

Related Functions

- GpiAssociate
- GpiCreatePS
- GpiDestroyPS
- GpiQueryDevice
- GpiQueryPS
- GpiResetPS
- GpiRestorePS
- GpiSavePS

GpiSetPS - Example Code

In this example, GpiSetPS is used to reset the presentation space.

```
#include <OS2.H>
#define INCL_GPICONTROL

HPS    hps;        /* presentation space handle  */
PSIZEL psizlsize;  /* presentation space size    */
ULONG  flOptions;  /* reset options              */

flOptions = PU_ARBITRARY |  /* arbitrary units   */
            GPIF_DEFAULT;   /* normal ps format  */

GpiSetPS(hps, psizlsize, flOptions);
```

GpiSetPS - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetRegion

GpiSetRegion - Syntax

This function changes a region to be the logical-OR of a set of rectangles.

```
#define INCL_GPIREGIONS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;             /*  Presentation-space handle. */
HRGN      hrgn;            /*  Region handle. */
LONG      lcount;          /*  Count of rectangles. */
PRECTL    arclRectangles;  /*  Array of rectangles. */
BOOL      rc;              /*  Success indicator. */

rc = GpiSetRegion(hps, hrgn, lcount, arclRectangles);
```

GpiSetRegion Parameter - hps

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

GpiSetRegion Parameter - hrgn

hrgn (HRGN) - input
Region handle.

GpiSetRegion Parameter - lcount

lcount (LONG) - input
Count of rectangles.

This is the number of rectangles specified in *arclRectangles* . If *lcount* = 0, the region is set to EMPTY, and *arclRectangles* is ignored.

GpiSetRegion Parameter - arclRectangles

arclRectangles (PRECTL) - input
Array of rectangles.

The rectangles are specified in device coordinates.

For each rectangle in the array, the value of *xRight* must be greater than (or equal to) *xLeft*, and *yTop* must be greater than (or equal to) *yBottom*.

GpiSetRegion Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetRegion - Parameters

hps (HPS) - input
Presentation-space handle.

The region must be owned by the device identified by the currently associated device context.

hrgn (HRGN) - input
Region handle.

lcount (LONG) - input
Count of rectangles.

This is the number of rectangles specified in *arclRectangles*. If *lcount* = 0, the region is set to EMPTY, and *arclRectangles* is ignored.

arclRectangles (PRECTL) - input
Array of rectangles.

The rectangles are specified in device coordinates.

For each rectangle in the array, the value of *xRight* must be greater than (or equal to) *xLeft*, and *yTop* must be greater than (or equal to) *yBottom*.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetRegion - Remarks

This function is similar to GpiCreateRegion, except that it changes an already existing region to be the logical-OR of the supplied rectangles,

instead of creating a new region.

The previous contents of the region are irrelevant. Points on the right-hand and top boundaries are not included in the changed region; points on the left-hand and bottom boundaries, that are not also on the right-hand or top boundaries, (that is, the top-left and bottom-right corner points) are included.

It is invalid if the specified region is currently selected as the clip region (by GpiSetClipRegion).

GpiSetRegion - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_HRGN (0x2080)
An invalid region handle was specified.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
An invalid rectangle parameter was specified.

PMERR_REGION_IS_CLIP_REGION (0x20F8)
An attempt was made to perform a region operation on a region that is selected as a clip region.

PMERR_HRGN_BUSY (0x2034)
An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.

GpiSetRegion - Related Functions

Related Functions

- GpiCombineRegion
- GpiCreateRegion
- GpiDestroyRegion
- GpiEqualRegion
- GpiOffsetRegion
- GpiPaintRegion
- GpiPtInRegion
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiRectInRegion

GpiSetRegion - Example Code

In this example we change the region to be the logical-or of a set of rectangles.

```
#define INCL_GPIREGIONS
#include <OS2.H>
#define maxrects 2

BOOL flResult;      /* success indicator.        */
HPS hps;            /* presentation space handle. */
HRGN hrgn;          /* region handle.             */
RECTL arclRect[maxrects] = {{20l, 20L,
                             40L, 40L},
                            {40L, 20L,
                             60L, 40L}};

/* array of rectangle structures */

flResult =  GpiSetRegion(hps,
                         hrgn,
                         (LONG)maxrects,
                    /* array of two rectangles. */
                               arclRect);
```

GpiSetRegion - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetSegmentAttrs

GpiSetSegmentAttrs - Syntax

This function sets a segment attribute.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;         /*  Presentation-space handle. */
LONG    lSegid;      /*  Segment identifier. */
LONG    lAttribute;  /*  Segment attribute. */
LONG    lValue;      /*  Attribute value. */
BOOL    rc;          /*  Success indicator. */

rc = GpiSetSegmentAttrs(hps, lSegid, lAttribute,
       lValue);
```

GpiSetSegmentAttrs Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetSegmentAttrs Parameter - lSegid

lSegid (LONG) - input
Segment identifier.

The identifier of the segment whose attribute is to be updated. It must be greater than zero.

GpiSetSegmentAttrs Parameter - lAttribute

lAttribute (LONG) - input
Segment attribute.

For details of the following attributes, see the GpiSetInitialSegmentAttrs function.

ATTR_DETECTABLE
Detectability
ATTR_VISIBLE
Visibility
ATTR_CHAINED
Chained
ATTR_DYNAMIC
Dynamic
ATTR_FASTCHAIN
Fast chaining
ATTR_PROP_DETECTABLE
Propagate detectability
ATTR_PROP_VISIBLE
Propagate visibility.

GpiSetSegmentAttrs Parameter - lValue

lValue (LONG) - input
Attribute value.

ATTR_ON
On/yes
ATTR_OFF
Off/no.

GpiSetSegmentAttrs Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetSegmentAttrs - Parameters

hps (HPS) - input
Presentation-space handle.

lSegid (LONG) - input
Segment identifier.

The identifier of the segment whose attribute is to be updated. It must be greater than zero.

lAttribute (LONG) - input
Segment attribute.

For details of the following attributes, see the GpiSetInitialSegmentAttrs function.

ATTR_DETECTABLE
Detectability
ATTR_VISIBLE
Visibility
ATTR_CHAINED
Chained
ATTR_DYNAMIC
Dynamic
ATTR_FASTCHAIN
Fast chaining
ATTR_PROP_DETECTABLE
Propagate detectability
ATTR_PROP_VISIBLE
Propagate visibility.

lValue (LONG) - input
Attribute value.

ATTR_ON
On/yes
ATTR_OFF
Off/no.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetSegmentAttrs - Remarks

This function sets the value of one segment attribute for the specified segment. The segment can be any retained segment.

If the identifier is that of the currently-open segment:

- In **retain** mode, this is valid.
- In **draw-and-retain** mode, the retained segment is updated, but there is no change to the immediate drawing.
- In **draw** mode, it is invalid.

(For a description of drawing mode, see GpiSetDrawingMode).

When a segment is modified from nonchained to chained, it is added to the end of the drawing chain.

GpiSetSegmentAttrs - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_SEG_ATTR (0x20C5)
An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.

PMERR_INV_SEG_ATTR_VALUE (0x20C6)
An invalid attribute value parameter was specified with GpiSetSegmentAttrs or GpiSetInitialSegmentAttrs.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiSetSegmentAttrs - Related Functions

Related Functions

- GpiCallSegmentMatrix
- GpiCloseSegment
- GpiCorrelateSegment
- GpiDeleteSegment
- GpiDeleteSegments
- GpiDrawSegment
- GpiErrorSegmentData
- GpiOpenSegment
- GpiQuerySegmentAttrs
- GpiSetInitialSegmentAttrs
- GpiSetSegmentPriority

GpiSetSegmentAttrs - Example Code

This function is used to set the current value of the specified attribute.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

HPS hps;  /*   Presentation-space */
          /*   handle.            */
LONG lSegid;   /* Segment identifier; must      */
               /* be greater than 0.            */
               /*                               */
               /* The name of the               */
               /* segment for which             */
               /* attribute information is to   */
               /* be returned.                  */
LONG lAttribute; /* attribute to be queried     */
LONG lValue;
lAttribute = ATTR_VISIBLE;


lValue = GpiSetSegmentAttrs(hps,
                            lSegid,
                            lAttribute,
                            ATTR_ON);
```

GpiSetSegmentAttrs - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetSegmentPriority

GpiSetSegmentPriority - Syntax

This function changes the position of a segment within the segment chain, or adds a segment to the chain.

```
#define INCL_GPISEGMENTS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;        /*  Presentation-space handle. */
```

```
LONG    lSegid;     /*  Segment identifier. */
LONG    lRefSegid;  /*  Reference segment identifier. */
LONG    lOrder;     /*  Segment higher or lower. */
BOOL    rc;         /*  Success indicator. */

rc = GpiSetSegmentPriority(hps, lSegid, lRefSegid,
       lOrder);
```

GpiSetSegmentPriority Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetSegmentPriority Parameter - lSegid

lSegid (LONG) - input
Segment identifier.

The identifier of the segment whose priority is to be changed; it must be greater than 0.

GpiSetSegmentPriority Parameter - lRefSegid

lRefSegid (LONG) - input
Reference segment identifier.

The segment that identifies a position in the segment chain. The segment specified in the */Segid* parameter is placed either immediately before or after this segment, depending on the value specified in the */Order* parameter. Specifying 0 for */RefSegid* indicates that the position is to be the beginning or the end of the segment chain as defined by the value in the */Order* parameter.

GpiSetSegmentPriority Parameter - lOrder

lOrder (LONG) - input
Segment higher or lower.

Specifies whether the segment named in the */Segid* parameter is to be placed before or after the segment named in the */RefSegid* parameter. Possible values are:

LOWER_PRI
The segment named in the */Segid* parameter is to have a lower priority than the segment named in the */RefSegid* parameter. The */Segid* segment is placed before the */RefSegid* segment. If 0 is specified in the */RefSegid* parameter, the segment identified in the */Segid* parameter is placed as the highest priority segment.

HIGHER_PRI

The segment named in the *lSegid* parameter is to have a higher priority than the segment named in the *lRefSegid* parameter. The *lSegid* segment is placed after the *lRefSegid* segment. If 0 is specified in the *lRefSegid* parameter, the segment identified in the *lSegid* parameter is placed as the lowest priority segment.

GpiSetSegmentPriority Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetSegmentPriority - Parameters

hps (HPS) - input
Presentation-space handle.

lSegid (LONG) - input
Segment identifier.

The identifier of the segment whose priority is to be changed; it must be greater than 0.

lRefSegid (LONG) - input
Reference segment identifier.

The segment that identifies a position in the segment chain. The segment specified in the *lSegid* parameter is placed either immediately before or after this segment, depending on the value specified in the *lOrder* parameter. Specifying 0 for *lRefSegid* indicates that the position is to be the beginning or the end of the segment chain as defined by the value in the *lOrder* parameter.

lOrder (LONG) - input
Segment higher or lower.

Specifies whether the segment named in the *lSegid* parameter is to be placed before or after the segment named in the *lRefSegid* parameter. Possible values are:

LOWER_PRI
The segment named in the *lSegid* parameter is to have a lower priority than the segment named in the *lRefSegid* parameter. The *lSegid* segment is placed before the *lRefSegid* segment. If 0 is specified in the *lRefSegid* parameter, the segment identified in the *lSegid* parameter is placed as the highest priority segment.

HIGHER_PRI
The segment named in the *lSegid* parameter is to have a higher priority than the segment named in the *lRefSegid* parameter. The *lSegid* segment is placed after the *lRefSegid* segment. If 0 is specified in the *lRefSegid* parameter, the segment identified in the *lSegid* parameter is placed as the lowest priority segment.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetSegmentPriority - Remarks

The specified segment can be a segment that exists in the segment chain, or an unchained segment. The effect of this function on an unchained segment is to add it to the segment chain in the specified position.

The application may redraw the picture by drawing the segment chain (see GpiDrawChain). This causes the segments in the chain to be processed from beginning to end, so that if segments overlap, later ones are placed on top of earlier ones (assuming a default mix mode) and therefore appear to have higher priority. Changing the position of the segment in the chain therefore has the effect of changing its priority to the end user.

GpiSetSegmentPriority - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_ORDERING_PARM (0x20AB)
An invalid order parameter was specified with GpiSetSegmentPriority.

PMERR_SEG_AND_REFSEG_ARE_SAME (0x20FA)
The segid and refsegid specified with GpiSetSegmentPriority were the same.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiSetSegmentPriority - Related Functions

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiOpenSegment
- GpiQuerySegmentPriority

GpiSetSegmentPriority - Example Code

This example finds the segment with the highest priority and places a segment just before it with a lower priority.

```
#define INCL_GPISEGMENTS
#include <OS2.H>
```

```
HPS hps;         /*   Presentation-space */
                 /*   handle.            */
LONG lRefSegid;  /* Reference-segment           */
                 /* identifier.                */
LONG laddSegid = 20L;
LONG lSegid;

lSegid = GpiQuerySegmentPriority (hps,
                /* find the segment with the highest */
                /* priority.                        */
                                  0,
                                  HIGHER_PRI);

GpiSetSegmentPriority(hps,
                      lRefSegid,
                      laddSegid,
                      LOWER_PRI);
```

GpiSetSegmentPriority - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetSegmentTransformMatrix

GpiSetSegmentTransformMatrix - Syntax

This function sets the segment transform that normally applies to all of the primitives in the specified segment.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
LONG         lSegid;       /*  Segment identifier. */
LONG         lCount;       /*  Number of elements. */
PMATRIXLF    pmatlfarray;  /*  Transformation matrix. */
LONG         lOptions;     /*  Transform options. */
BOOL         rc;           /*  Success indicator. */

rc = GpiSetSegmentTransformMatrix(hps, lSegid,
       lCount, pmatlfarray, lOptions);
```

GpiSetSegmentTransformMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetSegmentTransformMatrix Parameter - lSegid

lSegid (LONG) - input
Segment identifier.

This must be greater than 0.

GpiSetSegmentTransformMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements to be used in the *pmatlfarray* parameter. If *lCount* is less than 9, the elements omitted default to the corresponding elements of the identity matrix (see below). Specifying *lCount* =0 denotes that the identity matrix is used.

GpiSetSegmentTransformMatrix Parameter - pmatlfarray

pmatlfarray (PMATRIXLF) - input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus, a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

GpiSetSegmentTransformMatrix Parameter - lOptions

lOptions (LONG) - input
Transform options.

Specifies how the existing segment transform is to be modified by the transform defined by the *pmatlfarray* parameter. The new

segment transform is computed, and the result stored back in the segment, replacing the existing value. When the segment is drawn, the stored segment transform is used to update the segment transform that is currently in effect, in an additive manner. Possible values are:

TRANSFORM_REPLACE
: The previous default segment transform is discarded and replaced by the specified transform.

TRANSFORM_ADD
: The specified transform is combined with the existing default segment transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms.

TRANSFORM_PREEMPT
: The specified transform is combined with the existing default segment transform, in the order (1) new transform, (2) existing transform.

GpiSetSegmentTransformMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetSegmentTransformMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lSegid (LONG) - input
Segment identifier.

This must be greater than 0.

lCount (LONG) - input
Number of elements.

The number of elements to be used in the *pmatlfarray* parameter. If *lCount* is less than 9, the elements omitted default to the corresponding elements of the identity matrix (see below). Specifying *lCount*=0 denotes that the identity matrix is used.

pmatlfarray (PMATRIXLF) - input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus, a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

lOptions (LONG) - input
Transform options.

Specifies how the existing segment transform is to be modified by the transform defined by the *pmatlfarray* parameter. The new segment transform is computed, and the result stored back in the segment, replacing the existing value. When the segment is drawn, the stored segment transform is used to update the segment transform that is currently in effect, in an additive manner. Possible values are:

TRANSFORM_REPLACE
: The previous default segment transform is discarded and replaced by the specified transform.

TRANSFORM_ADD
: The specified transform is combined with the existing default segment transform, in the order (1) existing transform, (2) new transform. This option is most useful for incremental updates to transforms.

TRANSFORM_PREEMPT
: The specified transform is combined with the existing default segment transform, in the order (1) new transform, (2) existing transform.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetSegmentTransformMatrix - Remarks

The matrix is used to update the segment transform of a retained segment, according to the value of the *lOptions* parameter.

The segment transform is actually a model transform that applies at the start of the segment. It can be overridden later in the segment with a GpiSetModelTransformMatrix function.

This function specifies the transform as a one-dimensional array of *lCount* elements, being the first *lCount* elements of a 3-row by 3-column matrix ordered in rows. The order of the elements is:

```
Matrix                  Array


a  b  0
c  d  0           (a,b,0,c,d,0,e,f,1)
e  f  1
```

The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point:

```
(a*x + c*y + e, b*x + d*y + f)
```

The initial value of the transform of a segment is the identity matrix, as shown below:

```
Matrix                  Array


1  0  0
0  1  0           (1,0,0,0,1,0,0,0,1)
0  0  1
```

If scaling values greater than unity are given (which only applies if the presentation space coordinate format, as set by the GpiCreatePS function, is GPIF_LONG) it is possible for the combined effect of this and any other relevant transforms to exceed fixed-point implementation limits. This causes an error.

Segment transforms do not apply to primitives outside segments.

GpiSetSegmentTransformMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_SEG_NAME (0x20C8)
An invalid segment identifier was specified.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_SEG_NOT_FOUND (0x2100)
The specified segment identifier did not exist.

PMERR_INV_MATRIX_ELEMENT (0x209B)
An invalid transformation matrix element was specified.

PMERR_INV_TRANSFORM_TYPE (0x20D0)
An invalid options parameter was specified with a transform matrix function.

GpiSetSegmentTransformMatrix - Related Functions

Related Functions

- GpiCallSegmentMatrix
- GpiQueryModelTransformMatrix
- GpiQuerySegmentTransformMatrix
- GpiSetModelTransformMatrix

GpiSetSegmentTransformMatrix - Example Code

This example sets the transformation matrix of the highest priority segment to scale everything by a factor of 2.

```
#define INCL_GPISEGMENTS
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */
LONG lSegid;     /*   Segment identifier. */

MATRIXLF matlfArray = {MAKEFIXED(2,0),
                       0,0,0,MAKEFIXED(2,0),
                       0,0,0,1};

                             /* array of Transform matrix  */
                             /* structures.                */

lSegid = GpiQuerySegmentPriority(hps,
                /* find the segment with the highest */
                /* priority.                         */
                                 0,
                                 HIGHER_PRI);
```

```
GpiSetSegmentTransformMatrix(hps,
                             lSegid,
                             9L,
                             &matlfArray);
```

GpiSetSegmentTransformMatrix - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetStopDraw

GpiSetStopDraw - Syntax

This function sets or clears the "stop draw" condition.

```
#define INCL_GPICONTROL /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lValue;  /*  Stop draw condition. */
BOOL    rc;      /*  Success indicator. */

rc = GpiSetStopDraw(hps, lValue);
```

GpiSetStopDraw Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetStopDraw Parameter - lValue

lValue (LONG) - input
Stop draw condition.

SDW_OFF
Clear the "stop draw" condition
SDW_ON
Set the "stop draw" condition.

GpiSetStopDraw Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetStopDraw - Parameters

hps (HPS) - input
Presentation-space handle.

lValue (LONG) - input
Stop draw condition.

SDW_OFF
Clear the "stop draw" condition
SDW_ON
Set the "stop draw" condition.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetStopDraw - Remarks

This function allows an application to set up, and control, an asynchronous thread on which long drawing operations may be performed. At the point at which the controlling thread stops a draw, it sets the "stop draw" condition. The controlling thread clears this condition after it has received an acknowledgment from the drawing thread.

The "stop draw" condition has no effect on any other calls.

If one of the following calls is made (or has already been initiated from another thread) to the same presentation space, the call is terminated if the "stop draw" condition exists:

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiPlayMetaFile
- GpiPutData.

The call terminates with a warning.

Any call other than GpiSetStopDraw, directed at a presentation space that is currently in use, gives a PMERR_PS_BUSY error condition.

Note: If this function is issued when an asynchronous draw to a metafile is taking place, the result is an unusable metafile.

GpiSetStopDraw - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_INV_STOP_DRAW_VALUE (0x20CF)
An invalid value parameter was specified with GpiSetStopDraw.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiSetStopDraw - Related Functions

Related Functions

- GpiDrawChain
- GpiDrawDynamics
- GpiDrawFrom
- GpiDrawSegment
- GpiPlayMetaFile
- GpiPutData
- GpiQueryStopDraw

GpiSetStopDraw - Example Code

This example shows how to stop drawing.

```
#define INCL_GPICONTROL
#include <OS2.H>
HPS hps;          /*   Presentation-space */
                  /*   handle.            */

GpiSetStopDraw(hps,SDW_OFF);
```

GpiSetStopDraw - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiSetTag

GpiSetTag - Syntax

This function specifies a tag by which the following primitives are to be known.

```
#define INCL_GPICORRELATION /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;   /*  Presentation-space handle. */
LONG    lTag;  /*  Tag identifier. */
BOOL    rc;    /*  Success indicator. */

rc = GpiSetTag(hps, lTag);
```

GpiSetTag Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetTag Parameter - lTag

lTag (LONG) - input
Tag identifier.

GpiSetTag Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetTag - Parameters

hps (HPS) - input
Presentation-space handle.

lTag (LONG) - input
Tag identifier.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetTag - Remarks

When GpiCorrelateChain, GpiCorrelateFrom, or GpiCorrelateSegment is used to locate an object, both the segment identifier and the primitive tag of the object are returned to the application program.

If a tag of 0 is specified, the primitives have no name and are not returned by the correlate call.

Initially, the default and current tag are 0. The default tag can be changed with GpiSetDefTag.

Primitives within an unnamed segment cannot be picked or correlated, and any tag applied to them is ignored.

This function is not allowed between GpiBeginArea and GpiEndArea calls, therefore, all primitives within an area have the same tag.

The attribute mode (see GpiSetAttrMode) determines whether the current value of the tag is preserved.

GpiSetTag - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)

An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
An attempt was made to issue a function that is invalid in a micro presentation space.

GpiSetTag - Related Functions

Related Functions

- GpiCorrelateChain
- GpiQueryDefTag
- GpiQueryTag
- GpiSetDefTag

GpiSetTag - Graphic Elements and Orders

Element Type: **OCODE_GSPIK**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Pick Identifier

Element Type: **OCODE_GPSPIK**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Pick Identifier

GpiSetTag - Example Code

This example opens a segment and calls GpiSetTag so that all of the following primitives will be associated with that tag.

```
#define INCL_GPICORRELATION
#include <OS2.H>
HPS hps;         /*   Presentation-space */
                 /*   handle.            */


GpiOpenSegment(hps, 0L);
GpiSetTag(hps, 0L);
```

GpiSetTag - Topics

Select an item:
Syntax
Parameters

Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetTextAlignment

GpiSetTextAlignment - Syntax

This function determines the alignment used to position the characters in a string.

```
#define INCL_GPIPRIMITIVES /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS     hps;     /*  Presentation-space handle. */
LONG    lHoriz;  /*  Horizontal alignment. */
LONG    lVert;   /*  Vertical alignment. */
BOOL    rc;      /*  Success indicator. */

rc = GpiSetTextAlignment(hps, lHoriz, lVert);
```

GpiSetTextAlignment Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetTextAlignment Parameter - lHoriz

lHoriz (LONG) - input
Horizontal alignment.

This parameter and the next one (*lVert*) specify the alignment of character strings horizontally and vertically. Together they define a reference point within the string that is positioned on the starting point specified for the string.

Note: The terms used in this definition (left, right, top and bottom) must be interpreted with regard to the direction of the current coordinate system, as follows:

Left	the side of the display corresponding to the lowest x-value.
Right	the side of the display corresponding to the highest x-value.
Top	the side of the display corresponding to the highest y-value.

Bottom the side of the display corresponding to the lowest y-value.

The possible values are:

TA_NORMAL_HORIZ
Normal alignment. This is the initial default. The alignment assumed depends on the current character direction as set by GpiSetCharDirection:

CHDIRN_LEFTRIGHT	Same as TA_LEFT.
CHDIRN_TOPBOTTOM	Same as TA_US.CENTER.
CHDIRN_RIGHTLEFT	Same as TA_RIGHT.
CHDIRN_BOTTOMTOP	Same as TA_CENTER.

TA_LEFT
Left alignment. The string is aligned on the left edge of its leftmost character.

TA_CENTER
Center alignment. The string is aligned on the arithmetic mean of Left and Right.

TA_RIGHT
Right alignment. The string is aligned on the right edge of its rightmost character.

TA_STANDARD_HORIZ
Standard alignment. This is the initial default. The alignment assumed depends on the current character direction:

CHDIRN_LEFTRIGHT	Same as TA_LEFT.
CHDIRN_TOPBOTTOM	Same as TA_US.LEFT.
CHDIRN_RIGHTLEFT	Same as TA_RIGHT.
CHDIRN_BOTTOMTOP	Same as TA_LEFT.

GpiSetTextAlignment Parameter - lVert

lVert (LONG) - input
Vertical alignment.

TA_NORMAL_VERT
Normal alignment. This is the initial default. The alignment assumed depends on the current character direction as set by GpiSetCharDirection:

CHDIRN_LEFTRIGHT	Same as TA_BASE.
CHDIRN_TOPBOTTOM	Same as TA_US.TOP.
CHDIRN_RIGHTLEFT	Same as TA_BASE.
CHDIRN_BOTTOMTOP	Same as TA_BOTTOM.

TA_TOP
Top alignment. The string is aligned on the top edge of its topmost character.

TA_HALF
Half alignment. The string is aligned on the arithmetic mean of Bottom and Top.

TA_BASE
Base alignment. The string is aligned on the base of its bottom character.

TA_BOTTOM
Bottom alignment. The string is aligned on the bottom edge of its bottom character.

TA_STANDARD_VERT
Standard alignment. This is the initial default. The alignment assumed depends on the current character direction:

CHDIRN_LEFTRIGHT	Same as TA_BOTTOM.
CHDIRN_TOPBOTTOM	Same as TA_US.TOP.
CHDIRN_RIGHTLEFT	Same as TA_BOTTOM.
CHDIRN_BOTTOMTOP	Same as TA_BOTTOM.

GpiSetTextAlignment Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetTextAlignment - Parameters

hps (HPS) - input
Presentation-space handle.

lHoriz (LONG) - input
Horizontal alignment.

This parameter and the next one (*lVert*) specify the alignment of character strings horizontally and vertically. Together they define a reference point within the string that is positioned on the starting point specified for the string.

Note: The terms used in this definition (left, right, top and bottom) must be interpreted with regard to the direction of the current coordinate system, as follows:

Left	the side of the display corresponding to the lowest x-value.
Right	the side of the display corresponding to the highest x-value.
Top	the side of the display corresponding to the highest y-value.
Bottom	the side of the display corresponding to the lowest y-value.

The possible values are:

TA_NORMAL_HORIZ
: Normal alignment. This is the initial default. The alignment assumed depends on the current character direction as set by GpiSetCharDirection:

CHDIRN_LEFTRIGHT	Same as TA_LEFT.
CHDIRN_TOPBOTTOM	Same as TA_US.CENTER.
CHDIRN_RIGHTLEFT	Same as TA_RIGHT.
CHDIRN_BOTTOMTOP	Same as TA_CENTER.

TA_LEFT
: Left alignment. The string is aligned on the left edge of its leftmost character.

TA_CENTER
: Center alignment. The string is aligned on the arithmetic mean of Left and Right.

TA_RIGHT
: Right alignment. The string is aligned on the right edge of its rightmost character.

TA_STANDARD_HORIZ
: Standard alignment. This is the initial default. The alignment assumed depends on the current character direction:

CHDIRN_LEFTRIGHT	Same as TA_LEFT.
CHDIRN_TOPBOTTOM	Same as TA_US.LEFT.
CHDIRN_RIGHTLEFT	Same as TA_RIGHT.
CHDIRN_BOTTOMTOP	Same as TA_LEFT.

lVert (LONG) - input
Vertical alignment.

TA_NORMAL_VERT
Normal alignment. This is the initial default. The alignment assumed depends on the current character direction as set by GpiSetCharDirection:

CHDIRN_LEFTRIGHT	Same as TA_BASE.
CHDIRN_TOPBOTTOM	Same as TA_US.TOP.
CHDIRN_RIGHTLEFT	Same as TA_BASE.
CHDIRN_BOTTOMTOP	Same as TA_BOTTOM.

TA_TOP
Top alignment. The string is aligned on the top edge of its topmost character.

TA_HALF
Half alignment. The string is aligned on the arithmetic mean of Bottom and Top.

TA_BASE
Base alignment. The string is aligned on the base of its bottom character.

TA_BOTTOM
Bottom alignment. The string is aligned on the bottom edge of its bottom character.

TA_STANDARD_VERT
Standard alignment. This is the initial default. The alignment assumed depends on the current character direction:

CHDIRN_LEFTRIGHT	Same as TA_BOTTOM.
CHDIRN_TOPBOTTOM	Same as TA_US.TOP.
CHDIRN_RIGHTLEFT	Same as TA_BOTTOM.
CHDIRN_BOTTOMTOP	Same as TA_BOTTOM.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetTextAlignment - Remarks

This function must not be issued in an area bracket. The attribute mode determines whether the current value of the text alignment attribute is preserved.

Support for this function is device dependent.

GpiSetTextAlignment - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_CHAR_ALIGN_ATTR (0x2117)
The text alignment attribute specified in GpiSetTextAlignment is not valid.

GpiSetTextAlignment - Related Functions

Related Functions

- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiPop
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryTextAlignment
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetBackColor
- GpiSetBackMix
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetMix
- GpiSetDefAttrs

GpiSetTextAlignment - Graphic Elements and Orders

Element Type: **OCODE_GSTA**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Text Alignment

Element Type: **OCODE_GPSTA**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Text Alignment

GpiSetTextAlignment - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Related Functions
Graphic Elements and Orders
Glossary

GpiSetViewingLimits

GpiSetViewingLimits - Syntax

This function establishes a clipping rectangle in model space.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS       hps;         /*  Presentation-space handle. */
PRECTL    prclLimits;  /*  Viewing limits in model space. */
BOOL      rc;          /*  Success indicator. */

rc = GpiSetViewingLimits(hps, prclLimits);
```

GpiSetViewingLimits Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetViewingLimits Parameter - prclLimits

prclLimits (PRECTL) - input
Viewing limits in model space.

GpiSetViewingLimits Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetViewingLimits - Parameters

hps (HPS) - input
Presentation-space handle.

prclLimits (PRECTL) - input
Viewing limits in model space.

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetViewingLimits - Remarks

Viewing limits can be set within a segment, and apply to all subsequent primitives in the segment and any segments it calls. They can be changed at any time within the segment and they are not subject to segment or model transformations. Limits specified in called segments override those set by the limits of the root segment.

The limits are reset to their default value at the start of each root segment, subject to the fast-chaining attribute, like primitive attributes. The initial default value is no clipping; this can be changed with GpiSetDefViewingLimits.

The boundaries are inclusive, so that points on them are not clipped (removed). If either the left boundary of *prclLimits* is greater than the right, or the bottom greater than the top, a NULL rectangle is defined. All points are clipped.

Attribute mode (see GpiSetAttrMode) has no effect on this function.

The viewing limits are converted under the current viewing and default viewing transformations to a clipping rectangle in the page. This remains in force until changed by a subsequent GpiSetViewingLimits function. Clipping actually takes place to the intersection of the viewing limits, the clip path, the clip region, the graphics field, and the client area on the device.

GpiSetViewingLimits - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

GpiSetViewingLimits - Related Functions

Related Functions

- GpiQueryViewingLimits
- GpiSetDefViewingLimits
- GpiSetGraphicsField

GpiSetViewingLimits - Graphic Elements and Orders

Element Type: **OCODE_GSVW**
This element type is generated if the attribute mode (see GpiSetAttrMode) is set to AM_NOPRESERVE.

Order: Set Viewing Window

Element Type: **OCODE_GPSVW**
This element type is generated if the attribute mode is set to AM_PRESERVE.

Order: Push and Set Viewing Window

GpiSetViewingLimits - Example Code

In this example the model space clipping region width is reduced to 400x400.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;         /*   Presentation-space */
                 /*   handle.            */

BOOL fSuccess;
RECTL rclLimits = {  /* viewing limits.    */
                  10,10,
                  410,410
                  };


fSuccess = GpiSetViewingLimits(hps,
                               &rclLimits);
```

GpiSetViewingLimits - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiSetViewingTransformMatrix

GpiSetViewingTransformMatrix - Syntax

This function sets the viewing transform that is to apply to any subsequently opened segments.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;          /*  Presentation-space handle. */
LONG         lCount;       /*  Number of elements. */
PMATRIXLF    pmatlfArray;  /*  Transformation matrix. */
LONG         lOptions;     /*  Transform option. */
BOOL         rc;           /*  Success indicator. */

rc = GpiSetViewingTransformMatrix(hps, lCount,
       pmatlfArray, lOptions);
```

GpiSetViewingTransformMatrix Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiSetViewingTransformMatrix Parameter - lCount

lCount (LONG) - input
Number of elements.

The number of elements supplied in *pmatlfArray* , that are to be examined, starting from the beginning of the structure. If *lCount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. Specifying *lCount* = 0 means that the identity matrix is used.

GpiSetViewingTransformMatrix Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

GpiSetViewingTransformMatrix Parameter - lOptions

lOptions (LONG) - input
Transform option.

Specifies how the specified transform is to be used to modify the existing viewing transform. This must be:

TRANSFORM_REPLACE
New and replace. The previous viewing transform is discarded and replaced by the specified transform.

GpiSetViewingTransformMatrix Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiSetViewingTransformMatrix - Parameters

hps (HPS) - input
Presentation-space handle.

lCount (LONG) - input
Number of elements.

The number of elements supplied in *pmatlfArray* , that are to be examined, starting from the beginning of the structure. If *lCount* is less than 9, remaining elements default to the corresponding elements of the identity matrix. Specifying *lCount* = 0 means that the identity matrix is used.

pmatlfArray (PMATRIXLF) - input
Transformation matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers. If the presentation space coordinate format is GPIF_SHORT (see GpiCreatePS), these elements must be within the range -1 through +1.

The third, sixth, and ninth elements, when specified, must be 0, 0, and 1, respectively.

lOptions (LONG) - input
Transform option.

Specifies how the specified transform is to be used to modify the existing viewing transform. This must be:

TRANSFORM_REPLACE
New and replace. The previous viewing transform is discarded and replaced by the specified transform.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiSetViewingTransformMatrix - Remarks

This function is only valid outside segments. The viewing transform that is set applies to all subsequently opened (new) segments (it has no effect on primitives outside segments). All graphics primitives in a segment must have the same viewing transform. When it has been set for a particular segment, the viewing transform for that segment cannot be changed.

The transform is specified as a one-dimensional array of *lCount* elements, being the first n elements of a 3-row by 3-column matrix ordered by rows. The order of the elements is:

```
Matrix                  Array


a  b  0
c  d  0          (a,b,0,c,d,0,e,f,1)
e  f  1
```

The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point:

```
(a*x + c*y + e, b*x + d*y + f)
```

The initial value of the viewing transform is the identity matrix, as shown below:

```
Matrix                  Array


1  0  0
0  1  0          (1,0,0,0,1,0,0,0,1)
0  0  1
```

The viewing transform must be set (or defaulted) to the unity transform, before any segment that is to be called is first opened.

If scaling values greater than unity are given (which only applies if the presentation space coordinate format, as set by the GpiCreatePS function, is GPIF_LONG) it is possible for the combined effect of this and any other relevant transforms to exceed fixed-point implementation limits. This causes an error.

This function must not be issued in a path or area bracket.

GpiSetViewingTransformMatrix - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
: An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
: An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_MICROPS_FUNCTION (0x20A1)
: An attempt was made to issue a function that is invalid in a micro presentation space.

PMERR_INV_LENGTH_OR_COUNT (0x2092)

An invalid length or count parameter was specified.

PMERR_INV_MATRIX_ELEMENT (0x209B)
An invalid transformation matrix element was specified.

PMERR_INV_TRANSFORM_TYPE (0x20D0)
An invalid options parameter was specified with a transform matrix function.

PMERR_INV_IN_SEG (0x208D)
An attempt was made to issue a function invalid inside a segment bracket.

PMERR_NOT_IN_RETAIN_MODE (0x20E2)
An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.

GpiSetViewingTransformMatrix - Related Functions

Related Functions

- GpiQueryViewingTransformMatrix
- GpiSetDefaultViewMatrix

GpiSetViewingTransformMatrix - Example Code

In this example, the GpiSetViewingTransformMatrix is used to replace the existing viewing transformation. The new transformation will then double the width and height of drawing.

```
#define INCL_GPITRANSFORMS
#include <OS2.H>

HPS hps;          /* Presentation space handle. */
LONG lCount;      /* maximum number of elements */
MATRIXLF matlf = { MAKEFIXED(2,0),  /* scale x coordinates by a */
                                    /* factor of 2.             */
                   0, 0, 0,         /* no rotation.             */
                   MAKEFIXED(2,0),  /* scale y coordinates by a */
                                    /* factor of 2.             */
                   0, 0, 0, 1};     /* no rotation.             */

GpiSetViewingTransformMatrix(hps,
                             9L,    /* number of elements.      */
                             &matlf,
                             TRANSFORM_REPLACE);
```

GpiSetViewingTransformMatrix - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiStrokePath

GpiStrokePath - Syntax

This function strokes a path, and then draws it.

```
#define INCL_GPIPATHS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS      hps;        /*  Presentation-space handle. */
LONG     lPath;      /*  Identifier of path to be stroked; it must be 1. */
ULONG    flOptions;  /*  Stroke option. */
LONG     lHits;      /*  Correlation and error indicators. */

lHits = GpiStrokePath(hps, lPath, flOptions);
```

GpiStrokePath Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiStrokePath Parameter - lPath

lPath (LONG) - input
Identifier of path to be stroked; it must be 1.

GpiStrokePath Parameter - flOptions

flOptions (ULONG) - input
Stroke option.

Reserved; must be 0.

GpiStrokePath Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiStrokePath - Parameters

hps (HPS) - input
Presentation-space handle.

lPath (LONG) - input
Identifier of path to be stroked; it must be 1.

flOptions (ULONG) - input
Stroke option.

Reserved; must be 0.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
Successful
GPI_HITS
Correlate hits
GPI_ERROR
Error.

GpiStrokePath - Remarks

The path is first converted to one that describes the envelope of a wide line stroked using the current geometric line-width attribute (see GpiSetLineWidthGeom).

Note: This function and GpiModifyPath are the only calls that can cause geometric wide lines to be constructed. For more details about the way in which the envelope is constructed, see GpiModifyPath.

The converted path is then filled, using winding mode area fill and the area attributes. The boundaries of the wide line are included in the fill.

When it has been drawn, the path is deleted.

This function is equivalent to GpiModifyPath, followed by GpiFillPath. It is provided to enable device drivers to optimize storage, if possible.

If the current drawing mode (see GpiSetDrawingMode) is **draw** or **draw-and-retain**, drawing occurs on the currently associated device. If the drawing mode is **retain**, this function is stored in the current segment and output occurs when the segment is subsequently drawn in the usual way.

GpiStrokePath - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_PATH_ID (0x20AE)
An invalid path identifier parameter was specified.

PMERR_INV_RESERVED_FIELD (0x20C1)
An invalid reserved field was specified.

PMERR_PATH_UNKNOWN (0x20EE)
An attempt was made to perform a path function on a path that did not exist.

GpiStrokePath - Related Functions

Related Functions

- GpiBeginArea
- GpiBeginPath
- GpiEndPath
- GpiFillPath
- GpiModifyPath
- GpiOutlinePath
- GpiPathToRegion
- GpiSetAttrs
- GpiSetDefAttrs
- GpiSetClipPath
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetLineWidthGeom

GpiStrokePath - Graphic Elements and Orders

Element Type: **OCODE_GFPTH**
Note that GpiFillPath also generates this element type.

Order: Fill Path

GpiStrokePath - Example Code

This example uses the GpiStrokePath function to draw a wide line.

```
#define INCL_GPIPATHS
#include <OS2.H>
 HPS hps;          /* Presentation space handle. */
 POINTL ptlStart = { 0, 0 };
 POINTL ptlTriangle[] = { 100, 100, 200, 0, 0, 0 };

 /* create the path */

 GpiBeginPath(hps, 1L);
 GpiMove(hps, &ptlStart);
 GpiPolyLine(hps, 3, ptlTriangle);
 GpiEndPath(hps);

 GpiSetLineWidthGeom(hps, 20L); /* set the line width           */
 GpiStrokePath(hps, 1L, 0L);    /* draw the wide line           */
```

GpiStrokePath - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GpiTranslate

GpiTranslate - Syntax

This function applies a translation to a transform matrix.

```
#define INCL_GPITRANSFORMS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HPS          hps;              /*  Presentation-space handle. */
PMATRIXLF    pmatlfArray;      /*  Transform matrix. */
LONG         lOptions;         /*  Transform options. */
PPOINTL      pptlTranslation;  /*  Translation. */
BOOL         rc;               /*  Success indicator. */

rc = GpiTranslate(hps, pmatlfArray, lOptions,
       pptlTranslation);
```

GpiTranslate Parameter - hps

hps (HPS) - input
Presentation-space handle.

GpiTranslate Parameter - pmatlfArray

pmatlfArray (PMATRIXLF) - in/out
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

GpiTranslate Parameter - lOptions

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the specified translation should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE
The previous transform is discarded and replaced by the transform describing the specified translation.

TRANSFORM_ADD
The previous transform is combined with a transform representing the specified translation in the order (1) previous transform, (2) translational transform. This option is most useful for incremental updates to transforms.

GpiTranslate Parameter - pptlTranslation

pptlTranslation (PPOINTL) - input
Translation.

The coordinates of a point, relative to the origin, which defines the required translation.

GpiTranslate Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiTranslate - Parameters

hps (HPS) - input
Presentation-space handle.

pmatlfArray (PMATRIXLF) - in/out
Transform matrix.

The elements of the transform, in row order. The first, second, fourth, and fifth elements are of type FIXED, and have an assumed binary point between the second and third bytes. Thus a value of 1.0 is represented by 65 536. Other elements are normal signed integers.

The third, sixth, and ninth elements must be 0, 0, and 1, respectively.

lOptions (LONG) - input
Transform options.

Specifies how the transform defined by the specified translation should be used to modify the previous transform specified by the *pmatlfArray* parameter. Possible values are:

TRANSFORM_REPLACE
: The previous transform is discarded and replaced by the transform describing the specified translation.

TRANSFORM_ADD
: The previous transform is combined with a transform representing the specified translation in the order (1) previous transform, (2) translational transform. This option is most useful for incremental updates to transforms.

pptlTranslation (PPOINTL) - input
Translation.

The coordinates of a point, relative to the origin, which defines the required translation.

rc (BOOL) - returns
Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiTranslate - Remarks

This function is a helper function which either applies a specified translational component to an existing transform matrix, or replaces the matrix with one that represents the specified translation alone.

The transform is specified as a one-dimensional array of 9 elements that are the elements of a 3-row by 3-column matrix ordered by rows. The order of the elements are as follows:

```
Matrix                  Array


a  b  0
c  d  0            (a,b,0,c,d,0,e,f,1)
e  f  1
```

Transforms act on the coordinates of primitives, so that a point with coordinates (x,y) is transformed to the point:

```
(a*x + c*y + e, b*x + d*y + f)
```

The transform can be used in any call following:

- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix
- GpiSetDefaultViewMatrix.

Other similar helper functions are:

- GpiScale to apply a scaling component
- GpiRotate to apply a rotation component.

GpiTranslate - Errors

Possible returns from WinGetLastError

PMERR_INV_TRANSFORM_TYPE (0x20D0)
An invalid options parameter was specified with a transform matrix function.

GpiTranslate - Related Functions

Related Functions

- GpiRotate
- GpiScale
- GpiSetDefaultViewMatrix
- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiSetViewingTransformMatrix

GpiTranslate - Example Code

This example translates the center of the picture back to the center of the page.

```
#define INCL_GPITRANSFORMS
#define INCL_WINSYS
#include <OS2.H>

HPS hps;          /* Presentation space handle. */
MATRIXLF matlf;         /* Current viewing transformation */
POINTL ptlPictCenter;

/* determine the center of the page */
      ptlPictCenter.x = WinQuerySysValue(HWND_DESKTOP,
                           SV_CXFULLSCREEN)/2 - ptlPictCenter.x;
      ptlPictCenter.y = WinQuerySysValue(HWND_DESKTOP,
                           SV_CYFULLSCREEN)/2 - ptlPictCenter.y;
```

```
GpiQueryViewingTransformMatrix(hps,
                               9L,
/* Translate the center of the picture back to the center of the */
/* page.                                                         */
                               &matlf);
  GpiTranslate(hps,
               &matlf,
               TRANSFORM_ADD,
               &ptlPictCenter);
```

GpiTranslate - Topics

Select an item:
Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Glossary

GpiUnloadFonts

GpiUnloadFonts - Syntax

This function unloads any fonts previously loaded from the resource file by GpiLoadFonts.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB     hab;          /*  Anchor-block handle. */
PSZ     pszFilename;  /*  Fully qualified file name of the font resource. */
BOOL    rc;           /*  Success indicator. */

rc = GpiUnloadFonts(hab, pszFilename);
```

GpiUnloadFonts Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiUnloadFonts Parameter - pszFilename

pszFilename (PSZ) - input
Fully qualified file name of the font resource.

The file name extension is .FON

GpiUnloadFonts Return Value - rc

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiUnloadFonts - Parameters

hab (HAB) - input
Anchor-block handle.

pszFilename (PSZ) - input
Fully qualified file name of the font resource.

The file name extension is .FON

rc (BOOL) - returns
Success indicator.

TRUE
Successful completion
FALSE
Error occurred.

GpiUnloadFonts - Remarks

Before issuing this function, the application must:

1. Issue GpiSetCharSet to a font other than one of those to be unloaded, for example, to the default font.
2. Issue GpiDeleteSetId for each local identifier (lcid) that references one of the fonts (the LCID_ALL option can be used if all lcids are to be deleted).

An error is returned if lcids that reference one of the fonts still exist for this application, and a warning is logged if lcids exist for another application.

GpiUnloadFonts - Errors

Possible returns from WinGetLastError

PMERR_FONT_FILE_NOT_LOADED (0x202E)
An attempt was made to unload a font file that was not loaded.

PMERR_OWN_SET_ID_REFS (0x20EB)
An attempt to unload a font failed because the setid is still being referenced.

GpiUnloadFonts - Related Functions

Prerequisite Functions

- GpiDeleteSetId
- GpiSetCharSet

Related Functions

- GpiCreateLogFont
- GpiLoadFonts
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryKerningPairs
- GpiQueryNumberSetIds
- GpiQuerySetIds
- GpiQueryWidthTable

GpiUnloadFonts - Example Code

This function unloads any font(s) previously loaded from the resource file by GpiLoadFonts

```
#define INCL_GPILCIDS
#include <OS2.H>

HAB hab;          /* Anchor-block handle. */
char fntname[] = "HELVETICA.FON";


GpiUnloadFonts(hab, fntname);
```

GpiUnloadFonts - Topics

Select an item:
Syntax
Parameters
Returns

Errors
Remarks
Example Code
Related Functions
Glossary

GpiUnloadPublicFonts

GpiUnloadPublicFonts - Syntax

This function unloads one or more generally-available fonts from the specified resource file. See GpiLoadPublicFonts.

```
#define INCL_GPILCIDS /* Or use INCL_GPI, INCL_PM, */
#include <os2.h>

HAB     hab;          /*  Anchor-block handle. */
PSZ     pszFilename;  /*  Filename. */
BOOL    rc;           /*  Success indicator. */

rc = GpiUnloadPublicFonts(hab, pszFilename);
```

GpiUnloadPublicFonts Parameter - hab

hab (HAB) - input
Anchor-block handle.

GpiUnloadPublicFonts Parameter - pszFilename

pszFilename (PSZ) - input
Filename.

This is the fully-qualified name of the font resource. The file-name extension is .FON

GpiUnloadPublicFonts Return Value - rc

rc (BOOL) - returns

Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiUnloadPublicFonts - Parameters

hab (HAB) - input
: Anchor-block handle.

pszFilename (PSZ) - input
: Filename.

This is the fully-qualified name of the font resource. The file-name extension is .FON

rc (BOOL) - returns
: Success indicator.

TRUE
: Successful completion

FALSE
: Error occurred.

GpiUnloadPublicFonts - Remarks

Before issuing this function, the application must:

1. Issue GpiSetCharSet to a font other than one of those to be unloaded, for example, to the default font.
2. Issue GpiDeleteSetId for each local identifier (lcid) that references one of the fonts (the LCID_ALL option can be used if all lcids are to be deleted).

An error is returned if lcids that reference one of the fonts still exist for this or any other application, and the unload fails.

Note: If another application is using the fonts when this function is issued, so that the call fails, the fonts are likely to remain loaded until the next boot. This is true even if the other application has issued a GpiLoadPublicFonts, and will later issue a GpiUnloadPublicFonts, since the use count is not decremented when the call fails.

It is also possible for one application to get details of the fonts with GpiQueryFonts, but then to fail to be able to use them with GpiCreateLogFont, because another application unloaded them with GpiUnloadPublicFonts in the time between the two calls.

GpiUnloadPublicFonts - Errors

Possible returns from WinGetLastError

PMERR_FONT_FILE_NOT_LOADED (0x202E)
: An attempt was made to unload a font file that was not loaded.

PMERR_OWN_SET_ID_REFS (0x20EB)
: An attempt to unload a font failed because the setid is still being referenced.

GpiUnloadPublicFonts - Example Code

This function unloads one or more generally-available fonts from the specified resource file.

```
#define INCL_GPILCIDS
#include <OS2.H>

HAB hab;          /* Anchor-block handle. */
char fntname[] = "HELVETICA.FON";


GpiUnloadPublicFonts(hab,fntname);
```

GpiUnloadPublicFonts - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Glossary

GpiWCBitBlt

GpiWCBitBlt - Syntax

This function copies a rectangle of bit-map image data.

```
#define INCL_GPIBITMAPS /* Or use INCL_GPI, INCL_PM, Also in COMMON section */
#include <os2.h>

HPS        hpsTarget;   /*  Target presentation-space handle. */
HBITMAP    hbmSource;   /*  Source bit-map handle. */
LONG       lCount;      /*  Point count. */
PPOINTL    aptlPoints;  /*  Point array. */
LONG       lRop;        /*  Mixing function required. */
ULONG      flOptions;   /*  Options. */
LONG       lHits;       /*  Correlation and error indicators. */

lHits = GpiWCBitBlt(hpsTarget, hbmSource,
          lCount, aptlPoints, lRop, flOptions);
```

GpiWCBitBlt Parameter - hpsTarget

hpsTarget (HPS) - input
Target presentation-space handle.

GpiWCBitBlt Parameter - hbmSource

hbmSource (HBITMAP) - input
Source bit-map handle.

It is an error if this bit map is currently selected into a memory device context.

GpiWCBitBlt Parameter - lCount

lCount (LONG) - input
Point count.

This count must be equal to 4.

GpiWCBitBlt Parameter - aptlPoints

aptlPoints (PPOINTL) - input
Point array.

Array of *lCount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**. These are:

Tx1,Ty1
: Specify the bottom-left corner of the target rectangle in target world coordinates.

Tx2,Ty2
: Specify the top-right corner of the target rectangle in target world coordinates.

Sx1,Sy1
: Specify the bottom-left corner of the source rectangle in source device coordinates.

Sx2,Sy2
: Specify the top-right corner of the source rectangle in source device coordinates.

GpiWCBitBlt Parameter - lRop

lRop (LONG) - input
Mixing function required.

Each plane of the target can be considered to be processed separately. For any pel in a target plane, three bits together with the *lRop* values are used to determine the final value. These are the value of that pel in the Pattern (P) and Source (S) data and the initial value of that pel in the Target (T) data. For any combination of P, S, and T pel values, the final target value for the pel is determined by the appropriate *lRop* bit value as shown in the table below:

P	S	T (initial)	T (final)
0	0	0	Index bit 0 (least significant)
0	0	1	Index bit 1
0	1	0	Index bit 2
0	1	1	Index bit 3
1	0	0	Index bit 4
1	0	1	Index bit 5
1	1	0	Index bit 6
1	1	1	Index bit 7 (most significant)

The index formed in the above way determines the mixing required. Mnemonic names are available for commonly used mixes:

```
ROP_SRCCOPY       /* SRC                     */
ROP_SRCPAINT      /* SRC OR DST              */
ROP_SRCAND        /* SRC AND DST             */
ROP_SRCINVERT     /* SRC XOR DST             */
ROP_SRCERASE      /* SRC AND NOT(DST)        */
ROP_NOTSRCCOPY    /* NOT(SRC)                */
ROP_NOTSRCERASE   /* NOT(SRC) AND NOT(DST)   */
ROP_MERGECOPY     /* SRC AND PAT             */
ROP_MERGEPAINT    /* NOT(SRC) OR DST         */
ROP_PATCOPY       /* PAT                     */
ROP_PATPAINT      /* NOT(SRC) OR PAT OR DST  */
ROP_PATINVERT     /* DST XOR PAT             */
ROP_DSTINVERT     /* NOT(DST)                */
ROP_ZERO          /* 0                       */
ROP_ONE           /* 1                       */
```

GpiWCBitBlt Parameter - flOptions

flOptions (ULONG) - input
Options.

How eliminated lines or columns are treated if a compression is performed.

Flags 15 through 31 of *flOptions* can be used for privately-supported modes for particular devices.

BBO_OR
The default. If compression is necessary, logical-OR eliminated rows or columns. This is useful for white on black.

BBO_AND
If compression is necessary, logical-AND eliminated rows or columns. This is useful for black on white.

BBO_IGNORE
: If compression is necessary, ignore eliminated rows or columns. This is useful for color.

BBO_PAL_COLORS
: The color table passed in with the BitmapInfoTable is defined as indices into the currently realized palette, rather than actual colors.

GpiWCBitBlt Return Value - lHits

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiWCBitBlt - Parameters

hpsTarget (HPS) - input
Target presentation-space handle.

hbmSource (HBITMAP) - input
Source bit-map handle.

It is an error if this bit map is currently selected into a memory device context.

lCount (LONG) - input
Point count.

This count must be equal to 4.

aptlPoints (PPOINTL) - input
Point array.

Array of *lCount* points, in the order **Tx1, Ty1, Tx2, Ty2, Sx1, Sy1, Sx2, Sy2**. These are:

Tx1,Ty1
: Specify the bottom-left corner of the target rectangle in target world coordinates.

Tx2,Ty2
: Specify the top-right corner of the target rectangle in target world coordinates.

Sx1,Sy1
: Specify the bottom-left corner of the source rectangle in source device coordinates.

Sx2,Sy2
: Specify the top-right corner of the source rectangle in source device coordinates.

lRop (LONG) - input
Mixing function required.

Each plane of the target can be considered to be processed separately. For any pel in a target plane, three bits together with the *lRop* values are used to determine the final value. These are the value of that pel in the Pattern (P) and Source (S) data and the initial value of that pel in the Target (T) data. For any combination of P, S, and T pel values, the final target value for the pel is determined by the appropriate *lRop* bit value as shown in the table below:

```
P   S   T          T (final)
        (initial)

0   0   0          Index bit 0
                   (least
                   significant)

0   0   1          Index bit 1

0   1   0          Index bit 2

0   1   1          Index bit 3

1   0   0          Index bit 4

1   0   1          Index bit 5

1   1   0          Index bit 6

1   1   1          Index bit 7
                   (most
                   significant)
```

The index formed in the above way determines the mixing required. Mnemonic names are available for commonly used mixes:

```
ROP_SRCCOPY       /* SRC                      */
ROP_SRCPAINT      /* SRC OR DST               */
ROP_SRCAND        /* SRC AND DST              */
ROP_SRCINVERT     /* SRC XOR DST              */
ROP_SRCERASE      /* SRC AND NOT(DST)         */
ROP_NOTSRCCOPY    /* NOT(SRC)                 */
ROP_NOTSRCERASE   /* NOT(SRC) AND NOT(DST)    */
ROP_MERGECOPY     /* SRC AND PAT              */
ROP_MERGEPAINT    /* NOT(SRC) OR DST          */
ROP_PATCOPY       /* PAT                      */
ROP_PATPAINT      /* NOT(SRC) OR PAT OR DST   */
ROP_PATINVERT     /* DST XOR PAT              */
ROP_DSTINVERT     /* NOT(DST)                 */
ROP_ZERO          /* 0                        */
ROP_ONE           /* 1                        */
```

flOptions (ULONG) - input
Options.

How eliminated lines or columns are treated if a compression is performed.

Flags 15 through 31 of *flOptions* can be used for privately-supported modes for particular devices.

BBO_OR
: The default. If compression is necessary, logical-OR eliminated rows or columns. This is useful for white on black.

BBO_AND
: If compression is necessary, logical-AND eliminated rows or columns. This is useful for black on white.

BBO_IGNORE
: If compression is necessary, ignore eliminated rows or columns. This is useful for color.

BBO_PAL_COLORS
: The color table passed in with the BitmapInfoTable is defined as indices into the currently realized palette, rather than actual colors.

lHits (LONG) - returns
Correlation and error indicators.

GPI_OK
: Successful

GPI_HITS
: Correlate hits

GPI_ERROR
: Error.

GpiWCBitBlt - Remarks

A rectangle of bit-map image data is copied from a bit map, to a bit map selected into a device context associated with the target presentation space. Alternatively, the target presentation space can be associated with a device context that specifies a suitable raster device, for example, the screen.

Note: In either case, both source and target device contexts must apply to the same physical device. It is an error if this device does not support raster operations.

A rectangle is specified in device coordinates for the source bit map, and one in world coordinates for the target presentation space. The source rectangle is noninclusive; the left and lower boundaries in device space are included, but not the right and upper boundaries. Thus if the bottom-left is equal to the top-right, the rectangle is deemed to be empty. The target rectangle is "inclusive-inclusive"; that is, all boundaries are included in the rectangle.

If the target rectangle, after transformation to device coordinates and adjustment for inclusivity, is not the same size as the source rectangle, then stretching or compressing of the data occurs. *flOptions* specifies how eliminated rows or columns of bits are to be treated if compression occurs. Note that the pattern data is never stretched or compressed.

If there is a rotational effect in the transforms, the copy of the bit map is rotated accordingly.

The target rectangle is transformed to device coordinates, and if any shear or rotation has occurred, this is then converted to an upright rectangle that bounds the transformed figure. This rectangle is used as the target of the operation. No inversion of the image takes place.

These current attributes of the target presentation space are used (other than for converting between monochrome and color, as described below):

- Area color
- Area background color
- Pattern set
- Pattern symbol.

The color values are used in conversion between monochrome and color data. This is the only format conversion performed by this function. The conversions are:

- Output of a monochrome pattern to a color device.

 In this instance the pattern is converted first to a color pattern, using the current area colors:

 - source 1s -> area foreground color
 - source 0s -> area background color.

- Copying from a monochrome bit map to a color bit map (or device).

 The source bits are converted as follows:

 - source 1s -> image foreground color
 - source 0s -> image background color.

- Copying from a color bit map to a monochrome bit map (or device).

 The source bits are converted as follows:

 - source nonzeros -> image foreground color
 - source 0s -> image background color.

If the mix (*lRop*) does not call for a pattern, the pattern set and pattern symbol are not used.

Neither the source nor the pattern is required when a bit map, or part of a bit map, is to be cleared to a particular color.

If the mix does require both source and pattern, a three-way operation is performed.

If a pattern is required, dithering may be performed for solid patterns in a color that is not available on the device. See GpiSetPattern.

Support for the BM_SRCTRANSPARENT and BM_DESTTRANSPARENT background mix options is provided by the CAPS_BACKGROUND_MIX_SUPPORT flags in DevQueryCaps.

Requesting the BM_SRCTRANSPARENT background mix option results in pels from the source bit map matching the presentation space

background color to not be copied to the output bit map, effectively leaving those pels in the output unchanged. This provides for a transparent overlay function.

Requesting the BM_DESTTRANSPARENT background mix option results in a transfer such that pels from the source bit map will only be copied to destination pels that match the presentation space background color. This provides for a transparent underlay function.

This function (unlike GpiBitBlt) can be drawn immediately, retained in segment store, or both of these, depending upon the drawing mode (see GpiSetDrawingMode).

Note: There are restrictions on the use of this function when creating SAA-conforming metafiles; see "MetaFile Resolutions" in the *Presentation Manager Programming Reference* for more information.

GpiWCBitBlt - Errors

Possible returns from WinGetLastError

PMERR_INV_HPS (0x207F)
An invalid presentation-space handle was specified.

PMERR_PS_BUSY (0x20F4)
An attempt was made to access the presentation space from more than one thread simultaneously.

PMERR_INV_LENGTH_OR_COUNT (0x2092)
An invalid length or count parameter was specified.

PMERR_INV_BITBLT_MIX (0x2046)
An invalid *lRop* was specified with a GpiBitBlt or GpiWCBitBlt function.

PMERR_INV_BITBLT_STYLE (0x2047)
An invalid options parameter was specified with a GpiBitBlt or GpiWCBitBlt function.

PMERR_BITMAP_NOT_FOUND (0x200A)
A attempt was made to perform a bit-map operation on a bit map that did not exist.

PMERR_INV_COORDINATE (0x205B)
An invalid coordinate value was specified.

PMERR_INV_RECT (0x20BD)
An invalid rectangle parameter was specified.

PMERR_NO_BITMAP_SELECTED (0x20E4)
An attempt has been made to operate on a memory device context that has no bit map selected.

PMERR_INCORRECT_DC_TYPE (0x203C)
An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.

PMERR_INCOMPATIBLE_BITMAP (0x203A)
An attempt was made to select a bit map or perform a BitBlt operation on a device context that was incompatible with the format of the bit map.

PMERR_INV_HBITMAP (0x207B)
An invalid bit-map handle was specified.

PMERR_HBITMAP_BUSY (0x2032)
An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.

GpiWCBitBlt - Related Functions

Related Functions

- GpiBitBlt
- GpiCreateBitmap
- GpiDeleteBitmap
- GpiDrawBits
- GpiLoadBitmap
- GpiQueryBitmapBits
- GpiQueryBitmapDimension
- GpiQueryBitmapHandle
- GpiQueryBitmapParameters
- GpiQueryDeviceBitmapFormats
- GpiSetBitmap
- GpiSetBitmapBits
- GpiSetBitmapDimension
- GpiSetBitmapId

GpiWCBitBlt - Graphic Elements and Orders

Element Type: **OCODE_GBBLT**

Order: Bitblt

GpiWCBitBlt - Example Code

This function copies a rectangle of bit-map image data. This example uses GpiWCBitBlt to copy and compress a bit map in a presentation space. The function copies the bit map that is 100 pels wide and 100 pels high into a 50-by-50-pel rectangle at the location (300,400). Since the raster operation is ROP_SRCCOPY, GpiWCBitBlt replaces the image previously in the presentation-space rectangle. The function compresses the bit map to fit the new rectangle by discarding extra rows and columns as specified by the BBO_IGNORE option.

```
#define INCL_GPIBITMAPS
#include <OS2.H>
HPS hps;          /* Presentation space handle. */
HBITMAP hbm;
POINTL aptl[4] = {
     300, 400,      /* lower-left corner of target            */
     350, 450,      /* upper-right corner of target           */
     0, 0,          /* lower-left corner of source            */
     100, 100 };    /* upper-right corner of source           */

 GpiWCBitBlt(hps,   /* presentation space                     */
     hbm,           /* bit-map handle                          */
     4L,            /* four points needed to compress         */
     aptl,          /* points for source and target rectangles */
     ROP_SRCCOPY,   /* copy source replacing target           */
     BBO_IGNORE);   /* discard extra rows and columns         */
```

GpiWCBitBlt - Topics

Select an item:

Syntax
Parameters
Returns
Errors
Remarks
Example Code
Related Functions
Graphic Elements and Orders
Glossary

GPI Function Context

A number of the GPI error conditions indicate that a function has been used in the wrong context. This appendix lists every function and shows, for each one, whether it can be used:

- In a micro presentation space
- While there is an open segment bracket
- While there is an open area bracket
- While there is an open element bracket
- While there is an open path bracket.

A yes (Yes) means that a function can be used; a no (No) means that it cannot. There are some additional qualifiers in the form of superscript numbers. These generally indicate some further restriction on the context in which a function can be called, and are explained at the end of this appendix.

Where GPI Functions Can Be Called

Gpi Function	Micro PS	Segment Bracket	Area Bracket	Element Bracket	Path Bracket
GpiAnimatePalette	Yes(8,9)	Yes	No	Yes	No
GpiAssociate	No	Yes	No(7)	Yes(6)	No(7)
GpiBeginArea	Yes	Yes	No	Yes	No
GpiBeginElement	No	Yes	Yes	No	Yes
GpiBeginPath	Yes	Yes	No	Yes	No
GpiBitBlt	Yes	Yes	No	Yes	No
GpiBox	Yes	Yes	Yes	Yes	Yes
GpiCallSegmentMatrix	No	Yes	Yes	Yes	Yes
GpiCharString	Yes	Yes	No	Yes	Yes
GpiCharStringAt	Yes	Yes	No	Yes	Yes
GpiCharStringPos	Yes	Yes	No	Yes	Yes
GpiCharStringPosAt	Yes	Yes	No	Yes	Yes
GpiCloseFigure	Yes	Yes	No	Yes	Yes
GpiCloseSegment	No	Yes	No(7)	Yes	No(7)
GpiCombineRegion	Yes	Yes	No	Yes	No
GpiComment	Yes	Yes	Yes	Yes	Yes
GpiConvert	Yes	Yes	Yes	Yes	Yes
GpiConvertWithMatrix	Yes	Yes	Yes	Yes	Yes
GpiCopyMetaFile	Yes	Yes	Yes	Yes	Yes

GpiCorrelateChain	No	Yes	No	No	No
GpiCorrelateFrom	No	Yes	No	No	No
GpiCorrelateSegment	No	Yes	No	No	No
GpiCreateBitmap	Yes	Yes	Yes	Yes	Yes
GpiCreateLogColorTable	Yes	Yes	No	Yes	No
GpiCreateLogFont	Yes	Yes	No	No	Yes
GpiCreatePalette	Yes	Yes	Yes	Yes	Yes
GpiCreatePS	-	-	-	-	-
GpiCreateRegion	Yes	Yes	No	Yes	No
GpiDeleteBitmap	Yes	Yes	Yes	Yes	Yes
GpiDeleteElement	No	Yes(3)	Yes(3)	No	Yes(3)
GpiDeleteElementRange	No	Yes(3)	Yes(3)	No	Yes(3)
GpiDeleteElementsBetweenLabels	No	Yes(3)	Yes(3)	No	Yes(3)
GpiDeleteMetaFile	Yes	Yes	Yes	Yes	Yes
GpiDeletePalette	Yes(9)	Yes	Yes	Yes	Yes
GpiDeleteSegment	No	Yes	Yes	Yes	Yes
GpiDeleteSegments	No	Yes	Yes	Yes	Yes
GpiDeleteSetId	Yes	Yes	No	Yes	Yes
GpiDestroyPS	Yes(1)	Yes	Yes	Yes	Yes
GpiDestroyRegion	Yes	Yes	No	Yes	No
GpiDrawBits	Yes(10)	Yes	Yes	Yes	Yes
GpiDrawChain	No	Yes	No	No	No
GpiDrawDynamics	No	Yes	No	No	No
GpiDrawFrom	No	Yes	No	No	No
GpiDrawSegment	No	Yes	No	No	No
GpiElement	No	Yes	Yes	No	Yes
GpiEndArea	Yes	Yes	Yes	Yes	No
GpiEndElement	No	Yes	Yes	Yes	Yes
GpiEndPath	Yes	Yes	No	Yes	Yes
GpiEqualRegion	Yes	Yes	No	Yes	No
GpiErase	Yes	Yes	No	Yes	No
GpiErrorSegmentData	No	Yes	Yes	Yes	Yes
GpiExcludeClipRectangle	Yes	Yes	No	Yes	No
GpiFillPath	Yes	Yes	No	Yes	No
GpiFullArc	Yes	Yes	Yes	Yes	Yes
GpiGetData	No	Yes(5)	Yes	Yes	Yes
GpiImage	Yes	Yes	No	Yes	No
GpiIntersectClipRectangle	Yes	Yes	No	Yes	No
GpiLabel	No	Yes	Yes	No	Yes
GpiLine	Yes	Yes	Yes	Yes	Yes

GpiLoadBitmap	Yes	Yes	Yes	Yes	Yes
GpiLoadFonts	Yes	Yes	Yes	Yes	Yes
GpiLoadMetaFile	Yes	Yes	Yes	Yes	Yes
GpiLoadPublicFonts	Yes	Yes	Yes	Yes	Yes
GpiMarker	Yes	Yes	No	Yes	Yes
GpiModifyPath	Yes	Yes	No	Yes	No
GpiMove	Yes	Yes	Yes	Yes	Yes
GpiOffsetClipRegion	Yes	Yes	No	Yes	No
GpiOffsetElementPointer	No	Yes(3)	Yes(3)	No	Yes(3)
GpiOffsetRegion	Yes	Yes	No	Yes	No
GpiOpenSegment	No	No	No(7)	Yes(6)	No(7)
GpiOutlinePath	Yes	Yes	No	Yes	No
GpiPaintRegion	Yes	Yes	No	Yes	No
GpiPartialArc	Yes	Yes	Yes	Yes	Yes
GpiPathToRegion	Yes	Yes	Yes	Yes	No
GpiPlayMetaFile	Yes	No	No	No	No
GpiPointArc	Yes	Yes	Yes	Yes	Yes
GpiPolyFillet	Yes	Yes	Yes	Yes	Yes
GpiPolyFilletSharp	Yes	Yes	Yes	Yes	Yes
GpiPolygons	Yes	Yes	No	Yes	No
GpiPolyLine	Yes	Yes	Yes	Yes	Yes
GpiPolyLineDisjoint	Yes	Yes	Yes	Yes	Yes
GpiPolyMarker	Yes	Yes	No	Yes	Yes
GpiPolySpline	Yes	Yes	Yes	Yes	Yes
GpiPop	No	Yes	Yes	Yes	Yes
GpiPtInRegion	Yes	Yes	No	Yes	No
GpiPtVisible	Yes	Yes	No	Yes	No
GpiPutData	No	Yes	Yes	Yes	Yes
GpiQueryArcParams	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryAttrMode	No	Yes	Yes	Yes	Yes
GpiQueryAttrs	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryBackColor	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryBackMix	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryBitmapBits	Yes	Yes(4)	No	Yes	No
GpiQueryBitmapDimension	Yes	Yes	Yes	Yes	Yes
GpiQueryBitmapHandle	Yes	Yes	Yes	Yes	Yes
GpiQueryBitmapInfoHeader	Yes	Yes	Yes	Yes	Yes
GpiQueryBitmapParameters	Yes	Yes	Yes	Yes	Yes
GpiQueryBoundaryData	Yes	Yes	Yes	Yes	Yes

GpiQueryCharAngle	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryCharBox	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryCharBreakExtra	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryCharDirection	Yes	Yes	No	Yes	Yes
GpiQueryCharExtra	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryCharMode	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryCharSet	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryCharShear	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryCharStringPos	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryCharStringPosAt	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryClipBox	Yes	Yes	Yes	Yes	Yes
GpiQueryClipRegion	Yes	Yes	Yes	Yes	Yes
GpiQueryColor	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryColorData	Yes	Yes	Yes	Yes	Yes
GpiQueryColorIndex	Yes	Yes	Yes	Yes	Yes
GpiQueryCp	Yes	Yes	Yes	Yes	Yes
GpiQueryCurrentPosition	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryDefArcParams	Yes	Yes	Yes	Yes	Yes
GpiQueryDefAttrs	Yes	Yes	Yes	Yes	Yes
GpiQueryDefaultViewMatrix	Yes	Yes	Yes	Yes	Yes
GpiQueryDefCharBox	Yes	Yes	Yes	Yes	Yes
GpiQueryDefTag	Yes	Yes	Yes	Yes	Yes
GpiQueryDefViewingLimits	Yes	Yes	Yes	Yes	Yes
GpiQueryDevice	Yes	Yes	Yes	Yes	Yes
GpiQueryDeviceBitmapFormats	Yes	Yes	Yes	Yes	Yes
GpiQueryDrawControl	Yes	Yes	Yes	Yes	Yes
GpiQueryDrawingMode	No	Yes	Yes	Yes	Yes
GpiQueryEditMode	No	Yes	Yes	Yes	Yes
GpiQueryElement	No	Yes(3)	Yes(3)	Yes(3)	Yes(3)
GpiQueryElementPointer	No	Yes(3)	Yes(3)	Yes(3)	Yes(3)
GpiQueryElementType	No	Yes(3)	Yes(3)	Yes(3)	Yes(3)
GpiQueryFaceString	Yes	Yes	Yes	Yes	Yes
GpiQueryFontAction	Yes	Yes	Yes	Yes	Yes
GpiQueryFontMetrics	Yes	Yes	Yes	Yes	Yes
GpiQueryFonts	Yes	Yes	Yes	Yes	Yes
GpiQueryFullFontFileDescs	Yes	Yes	Yes	Yes	Yes
GpiQueryGraphicsField	Yes	Yes	Yes	Yes	Yes
GpiQueryInitialSegmentAttrs	No	Yes	Yes	Yes	Yes
GpiQueryKerningPairs	Yes	Yes	Yes	Yes	Yes
GpiQueryLineEnd	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)

GpiQueryLineJoin	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryLineType	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryLineWidth	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryLineWidthGeom	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryLogColorTable	Yes	Yes	Yes	Yes	Yes
GpiQueryLogicalFont	Yes	Yes	Yes	Yes	Yes
GpiQueryMarker	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)	
GpiQueryMarkerSet	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryMetaFileBits	Yes	Yes	Yes	Yes	Yes
GpiQueryMetaFileLength	Yes	Yes	Yes	Yes	Yes
GpiQueryMix	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryModelTransformMatrix	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryNearestColor	Yes	Yes	Yes	Yes	Yes
GpiQueryNumberSetIds	Yes	Yes	Yes	Yes	Yes
GpiQueryPageViewport	Yes	Yes	Yes	Yes	Yes
GpiQueryPalette	Yes	Yes	Yes	Yes	Yes
GpiQueryPaletteInfo	Yes	Yes	Yes	Yes	Yes
GpiQueryPattern	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryPatternRefPoint	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryPatternSet	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryPel	Yes	Yes	Yes	Yes	Yes
GpiQueryPickAperturePosition	Yes	Yes	Yes	Yes	Yes
GpiQueryPickApertureSize	Yes	Yes	Yes	Yes	Yes
GpiQueryPS	Yes	Yes	Yes	Yes	Yes
GpiQueryRealColors	Yes	Yes	Yes	Yes	Yes
GpiQueryRegionBox	Yes	Yes	Yes	Yes	Yes
GpiQueryRegionRects	Yes	Yes	Yes	Yes	Yes
GpiQueryRGBColor	Yes	Yes	Yes	Yes	Yes
GpiQuerySegmentAttrs	No	Yes	Yes	Yes	Yes
GpiQuerySegmentNames	No	Yes	Yes	Yes	Yes
GpiQuerySegmentPriority	No	Yes	Yes	Yes	Yes
GpiQuerySegmentTransformMatrix	No	Yes	Yes	Yes	Yes
GpiQuerySetIds	Yes	Yes	Yes	Yes	Yes
GpiQueryStopDraw	No	Yes	Yes	Yes	Yes
GpiQueryTag	No	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryTextAlignment	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryTextBox	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)
GpiQueryViewingLimits	Yes	Yes(2)	Yes(2)	Yes(2)	Yes(2)

GpiQueryViewingTransformMatrix	No	Yes	Yes	Yes	Yes
GpiQueryWidthTable	Yes	Yes	Yes	Yes	Yes
GpiRectInRegion	Yes	Yes	No	Yes	No
GpiRectVisible	Yes	Yes	No	Yes	No
GpiRemoveDynamics	No	Yes	No	No	No
GpiResetBoundaryData	Yes	Yes	Yes	Yes	Yes
GpiResetPS	Yes	Yes	Yes(6)	Yes(6)	Yes(6)
GpiRestorePS	Yes	Yes(4)	No	Yes(4)	No
GpiRotate	Yes	Yes	Yes	Yes	Yes
GpiSaveMetaFile	Yes	Yes	Yes	Yes	Yes
GpiSavePS	Yes	Yes(4)	No	Yes(4)	No
GpiScale	Yes	Yes	Yes	Yes	Yes
GpiSelectPalette	Yes	Yes	No	Yes	Yes
GpiSetArcParams	Yes	Yes	Yes	Yes	Yes
GpiSetAttrMode	No	Yes	Yes	Yes	Yes
GpiSetAttrs	Yes	Yes	No	Yes	Yes
GpiSetBackColor	Yes	Yes	No	Yes	No
GpiSetBackMix	Yes	Yes	No	Yes	No
GpiSetBitmap	Yes	Yes	No	Yes	No
GpiSetBitmapBits	Yes	Yes	No	Yes	No
GpiSetBitmapDimension	Yes	Yes	Yes	Yes	Yes
GpiSetBitmapId	Yes	Yes	Yes	Yes	Yes
GpiSetCharAngle	Yes	Yes	No	Yes	Yes
GpiSetCharBox	Yes	Yes	No	Yes	Yes
GpiSetCharBreakExtra	Yes	Yes	Yes	Yes	Yes
GpiSetCharDirection	Yes	Yes	No	Yes	Yes
GpiSetCharExtra	Yes	Yes	Yes	Yes	Yes
GpiSetCharMode	Yes	Yes	No	Yes	Yes
GpiSetCharSet	Yes	Yes	No	Yes	Yes
GpiSetCharShear	Yes	Yes	No	Yes	Yes
GpiSetClipPath	Yes	Yes	No	Yes	No
GpiSetClipRegion	Yes	Yes	No	Yes	No
GpiSetColor	Yes	Yes	No	Yes	Yes
GpiSetCp	Yes	Yes	No	Yes	Yes
GpiSetCurrentPosition	Yes	Yes	Yes	Yes	Yes
GpiSetDefArcParams	Yes	Yes	Yes	Yes	Yes
GpiSetDefAttrs	Yes	Yes	Yes	Yes	Yes
GpiSetDefaultViewMatrix	Yes	Yes	No	Yes	No
GpiSetDefTag	Yes	Yes	Yes	Yes	Yes

GpiSetDefViewingLimits	Yes	Yes	Yes	Yes	Yes
GpiSetDrawControl	Yes	No	No	No	No
GpiSetDrawingMode	No	No	No	No	No
GpiSetEditMode	No	Yes	Yes	No	Yes
GpiSetElementPointer	No	Yes(2)	Yes(2)	No	Yes(2)
GpiSetElementPointerAtLabel	No	Yes(2)	Yes(2)	No	Yes(2)
GpiSetGraphicsField	Yes	Yes	No	Yes	No
GpiSetInitialSegmentAttrs	No	Yes	Yes	Yes	Yes
GpiSetLineEnd	Yes	Yes	No	Yes	Yes
GpiSetLineJoin	Yes	Yes	No	Yes	Yes
GpiSetLineType	Yes	Yes	No	Yes	Yes
GpiSetLineWidth	Yes	Yes	No	Yes	Yes
GpiSetLineWidthGeom	Yes	Yes	No	Yes	No
GpiSetMarker	Yes	Yes	No	Yes	Yes
GpiSetMarkerBox	Yes	Yes	No	Yes	Yes
GpiSetMarkerSet	Yes	Yes	No	Yes	Yes
GpiSetMetaFileBits	Yes	Yes	Yes	Yes	Yes
GpiSetMix	Yes	Yes	No	Yes	Yes
GpiSetModelTransformMatrix	Yes	Yes	Yes	Yes	Yes
GpiSetPageViewport	Yes	Yes	No	Yes	No
GpiSetPaletteEntries	Yes(8)	Yes	No	Yes	Yes
GpiSetPattern	Yes	Yes	No	Yes	No
GpiSetPatternRefPoint	Yes	Yes	No	Yes	No
GpiSetPatternSet	Yes	Yes	No	Yes	No
GpiSetPickAperturePosition	Yes	Yes	No	Yes	No
GpiSetPickApertureSize	Yes	Yes	No	Yes	No
GpiSetPS	Yes	Yes	No	Yes	No
GpiSetRegion	Yes	Yes	Yes	Yes	Yes
GpiSetSegmentAttrs	Yes	Yes	No	Yes	No
GpiSetSegmentPriority	No	Yes	Yes	Yes	Yes
GpiSetSegmentTransformMatrix	No	No(5)	Yes	No	Yes
GpiSetStopDraw	No	Yes	Yes	Yes	Yes
GpiSetTag	No	Yes	Yes	Yes	Yes
GpiSetTextAlignment	No	Yes	No	Yes	Yes
GpiSetViewingLimits	Yes	Yes	Yes	Yes	Yes
GpiSetViewingTransformMatrix	Yes	Yes	No	Yes	No
GpiStrokePath	No	No	No	Yes	No
GpiTranslate	Yes	Yes	No	Yes	No
GpiUnloadFonts	Yes	Yes	Yes	Yes	Yes
GpiUnloadPublicFonts	Yes	Yes	Yes	Yes	Yes

GpiUnloadPublicFonts	Yes	Yes	Yes	Yes	Yes
GpiWCBitBlt	Yes	Yes	Yes	Yes	Yes

Notes:

1. Not valid to a cached micro presentation space.
2. Valid only when the actual drawing mode is draw or draw-and-retain. The actual drawing mode is determined as shown in the following table:

 The Current Drawing Mode

GpiSetDrawingMode parameter	Context		
| | Chained Segment | Unchained Segment | Outside Segment |
 | DM_DRAWANDRETAIN | draw-and-retain | retain | draw |
 | DM_RETAIN | retain | retain | draw |
 | DM_DRAW | draw | retain | draw |

 For example, if the current drawing mode parameter is DM_RETAIN, and primitives are being drawn outside a segment, then the actual drawing mode is draw.
3. Valid only when the actual drawing mode (see note 2) is retain.
4. Valid only when the actual drawing mode (see note 2) is draw.
5. Not valid if the specified segment is the current open segment.
6. Bracket (path, element, or area) is ended without error.
7. Severity is Warning.
8. If associated with a metafile, only the final values are recorded in the metafile.
9. Palette must not be current.
10. Device context must be able to support bit map operations.

Data Types

The following data types are used in Graphics Presentation Interface. They are listed in alphabetic order.

ARCPARAMS

Arc-parameters structure.

```
typedef struct _ARCPARAMS {
  LONG     lP;  /*  P coefficient. */
  LONG     lQ;  /*  Q coefficient. */
  LONG     lR;  /*  R coefficient. */
  LONG     lS;  /*  S coefficient. */
} ARCPARAMS;

typedef ARCPARAMS *PARCPARAMS;
```

ARCPARAMS Field - lP

lP (LONG)
P coefficient.

ARCPARAMS Field - lQ

lQ (LONG)
Q coefficient.

ARCPARAMS Field - lR

lR (LONG)
R coefficient.

ARCPARAMS Field - lS

lS (LONG)
S coefficient.

AREABUNDLE

Area-attributes bundle structure.

```
typedef struct _AREABUNDLE {
  LONG       lColor;         /*  Area foreground color. */
  LONG       lBackColor;     /*  Area background color. */
  USHORT     usMixMode;      /*  Area foreground-mix mode. */
  USHORT     usBackMixMode;  /*  Area background-mix mode. */
  USHORT     usSet;          /*  Pattern set. */
  USHORT     usSymbol;       /*  Pattern symbol. */
  POINTL     ptlRefPoint;    /*  Pattern reference point. */
} AREABUNDLE;

typedef AREABUNDLE *PAREABUNDLE;
```

AREABUNDLE Field - lColor

lColor (LONG)
Area foreground color.

AREABUNDLE Field - lBackColor

lBackColor (LONG)
Area background color.

AREABUNDLE Field - usMixMode

usMixMode (USHORT)
Area foreground-mix mode.

AREABUNDLE Field - usBackMixMode

usBackMixMode (USHORT)
Area background-mix mode.

AREABUNDLE Field - usSet

usSet (USHORT)
Pattern set.

AREABUNDLE Field - usSymbol

usSymbol (USHORT)
Pattern symbol.

AREABUNDLE Field - ptlRefPoint

ptlRefPoint (POINTL)
Pattern reference point.

BITMAPINFO

Bit-map information structure.

Each bit plane logically contains (*cx* * *cy* * *cBitCount*) bits, although the actual length can be greater because of padding.

See also BITMAPINFO2, which is preferred.

```
typedef struct _BITMAPINFO {
  ULONG      cbFix;         /*  Length of fixed portion of structure. */
  USHORT     cx;            /*  Bit-map width in pels. */
  USHORT     cy;            /*  Bit-map height in pels. */
  USHORT     cPlanes;       /*  Number of bit planes. */
  USHORT     cBitCount;     /*  Number of bits per pel within a plane. */
  RGB        argbColor[1];  /*  Array of RGB values. */
} BITMAPINFO;

typedef BITMAPINFO *PBITMAPINFO;
```

BITMAPINFO Field - cbFix

cbFix (ULONG)
Length of fixed portion of structure.

This length can be determined using sizeof(BITMAPINFOHEADER).

BITMAPINFO Field - cx

cx (USHORT)
Bit-map width in pels.

BITMAPINFO Field - cy

cy (USHORT)
Bit-map height in pels.

BITMAPINFO Field - cPlanes

cPlanes (USHORT)
Number of bit planes.

BITMAPINFO Field - cBitCount

cBitCount (USHORT)
Number of bits per pel within a plane.

BITMAPINFO Field - argbColor[1]

argbColor[1] (RGB)
Array of RGB values.

This is a packed array of 24-bit RGB values. If there are N bits per pel (N = *cPlanes* * *cBitCount*), the array contains 2**N RGB values. However, if N = 24, the bit map does not need the **color** *color* array because the standard-format bit map, with 24 bits per pel, is assumed to contain RGB values.

BITMAPINFO2

Bit-map information structure.

Each bit plane logically contains (*cx* * *cy* * *cBitCount*) bits, although the actual length can be greater because of padding.

Note: Many functions can accept either this structure or the BITMAPINFO structure. Where possible, BITMAPINFO2 should be used.

The *cbFix* field is used to find the color table, if any, that goes with the information in this structure. A color table is an array of color (RGB2) values. If there are N bits per pel (N = *cPlanes* * *cBitCount*), the array contains 2**N color values. However, if N = 24, the color table is not required (because the standard-format bit map, with 24 bits per pel, is assumed to contain RGB values), unless either *cclrUsed* or *cclrImportant* is non-zero.

```
typedef struct _BITMAPINFO2 {
  ULONG      cbFix;            /*  Length of fixed portion of structure. */
  ULONG      cx;               /*  Bit-map width in pels. */
  ULONG      cy;               /*  Bit-map height in pels. */
  USHORT     cPlanes;          /*  Number of bit planes. */
  USHORT     cBitCount;        /*  Number of bits per pel within a plane. */
  ULONG      ulCompression;    /*  Compression scheme used to store the bit map. */
  ULONG      cbImage;          /*  Length of bit-map storage data, in bytes. */
  ULONG      cxResolution;     /*  Horizontal component of the resolution of target device. */
  ULONG      cyResolution;     /*  Vertical component of the resolution of the target device. */
  ULONG      cclrUsed;         /*  Number of color indexes used. */
  ULONG      cclrImportant;    /*  Minimum number of color indexes for satisfactory appearance of the bit map.
  USHORT     usUnits;          /*  Units of measure. */
  USHORT     usReserved;       /*  Reserved. */
  USHORT     usRecording;      /*  Recording algorithm. */
  USHORT     usRendering;      /*  Halftoning algorithm. */
  ULONG      cSize1;           /*  Size value 1. */
  ULONG      cSize2;           /*  Size value 2. */
  ULONG      ulColorEncoding;  /*  Color encoding. */
  ULONG      ulIdentifier;     /*  Reserved for application use. */
  RGB2       argbColor[1];     /*  Array of RGB values. */
} BITMAPINFO2;

typedef BITMAPINFO2 *PBITMAPINFO2;
```

BITMAPINFO2 Field - cbFix

cbFix (ULONG)
Length of fixed portion of structure.

The structure can be truncated after *cBitCount* or any subsequent field.

The length does not include the length of the color table. Where the color table is present, it is at an offset of *cbFix* from the start of the BITMAPINFO2 structure.

This length can range from 16 (BITMAPINFOHEADER through field *cBitCount*) up to sizeof(BITMAPINFOHEADER2) bytes.

BITMAPINFO2 Field - cx

cx (ULONG)
Bit-map width in pels.

BITMAPINFO2 Field - cy

cy (ULONG)
Bit-map height in pels.

BITMAPINFO2 Field - cPlanes

cPlanes (USHORT)
Number of bit planes.

BITMAPINFO2 Field - cBitCount

cBitCount (USHORT)
Number of bits per pel within a plane.

BITMAPINFO2 Field - ulCompression

ulCompression (ULONG)
Compression scheme used to store the bit map.

BCA_UNCOMP
Bit map is uncompressed.

BCA_HUFFMAN1D
The bit map is compressed by a modified Huffman encoding. This is valid for a bi-level (one bit per pel) bit map.

BCA_RLE4
The bit map is a 4-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

BCA_RLE8
The bit map is an 8-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

BCA_RLE24
The bit map is a 24-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

Format of Compressed Data

Encoding a run length:

Run-length encoded bit maps are encoded in the buffer in a controlled format. In all cases, if the first byte is non-zero, it is the length of a run of pels of a particular color or, in the case of a BCA_RLE4 bit map, a run of a length of pels of alternating colors.

```
1st-byte       pel repetition count >= 1
2nd-4th bytes  (BCA_RLE24 only) RGB value of pel.
```

```
2nd-byte       (BCA_RLE8) color index of pel to be repeated
               (BCA_RLE4) the second byte contains 2 4-bit
               color indexes.  The repetition count is
               completed by alternately choosing the high-order
               nibble followed by the low-order nibble for the
               succeeding pels until the count is exhausted.
```

Unencoded run:

An unencoded run is a string of pels to be placed in consecutive positions in the destination bit map.

```
1st-byte     0
2nd-byte     COUNT = a multiple of 3 for BCA_RLE24 bit maps, or
             COUNT >= 3 (for BCA_RLE4 and BCA_RLE8 bit maps).
```

followed by the bytes as follows:

BCA_RLE24
: A string of bytes specifying the RGB color values of succeeding pels. If COUNT is odd, it must be padded by a zero byte for an even length overall.

BCA_RLE8
: A string of bytes specifying color indexes for succeeding pels. If COUNT is odd, it must be padded by a zero byte for an even length overall.

BCA_RLE4
: A string of bytes, each byte providing two color indexes, with the high-order nibble specifying the index of the pel preceding the low-order nibble. The COUNT specifies the number of indexes. The overall length of the string must be an even number of bytes, and thus may be padded with a zero byte, and the low order nibble of the last significant byte may also be zero and not used.

Delta record:

A delta record indicates a shift in position in the destination bit map before the next record is decoded.

```
1st-byte        0
2nd-byte        2
3rd-byte        Delta-x (unsigned)
4th-byte        Delta-y (unsigned)
```

This is a relative jump record. It implies that the next record is to be decoded into a position in the destination bit map at an offset from the current position, determined by changing the horizontal and vertical positions by Delta-x and Delta-y, respectively.

End-of-line record:

The end-of-line record signifies that the data for the current scan line is complete and that decoding of the next record should begin at the start of the next scan line.

```
1st-byte        0
2nd-byte        0
```

End-of-RLE record:

The end-of-RLE record signifies the end of the data in the RLE compressed bit map.

```
1st-byte        0
2nd-byte        1
```

BITMAPINFO2 Field - cbImage

cbImage (ULONG)
Length of bit-map storage data, in bytes.

If the bit map is uncompressed, zero (default) can be specified for this.

BITMAPINFO2 Field - cxResolution

cxResolution (ULONG)
Horizontal component of the resolution of target device.

The resolution of the device the bit map is intended for, in the units specified by *usUnits* . This information enables an application to select from a resource group the bit map that best matches the characteristics of the current output device.

BITMAPINFO2 Field - cyResolution

cyResolution (ULONG)
Vertical component of the resolution of the target device.

See the description of *cxResolution* .

BITMAPINFO2 Field - cclrUsed

cclrUsed (ULONG)
Number of color indexes used.

The number of color indexes from the color table that are used by the bit map. If it is zero (the default), all the indexes are used. If it is non-zero, only the first *cclrUsed* entries in the table are accessed by the system, and further entries can be omitted.

For the standard formats with a *cBitCount* of 1, 4, or 8 (and *cPlanes* equal to 1), any indexes beyond *cclrUsed* are not valid. For example, a bit map with 64 colors can use the 8-bitcount format without having to supply the other 192 entries in the color table. For the 24-bitcount standard format, *cclrUsed* is the number of colors used by the bit map.

BITMAPINFO2 Field - cclrImportant

cclrImportant (ULONG)

Minimum number of color indexes for satisfactory appearance of the bit map.

More colors may be used in the bit map, but it is not necessary to assign them to the device palette. These additional colors may be mapped to the nearest colors available.

Zero (the default) means that all entries are important.

For a 24-bitcount standard format bit map, the *cclrImportant* colors are also listed in the color table following the BITMAPINFO2 structure.

BITMAPINFO2 Field - usUnits

usUnits (USHORT)
Units of measure.

Units of measure of the horizontal and vertical components of resolution, *cxResolution* and *cyResolution*.

BRU_METRIC	Pels per meter. This is the default value.

BITMAPINFO2 Field - usReserved

usReserved (USHORT)
Reserved.

This is a reserved field.

BITMAPINFO2 Field - usRecording

usRecording (USHORT)
Recording algorithm.

The format in which the bit map data is recorded.

BRA_BOTTOMUP	Scan lines are recorded bottom to top. This is the default value.

BITMAPINFO2 Field - usRendering

usRendering (USHORT)
Halftoning algorithm.

The algorithm used to record bit map data that has been digitally halftoned.

BRH_NOTHALFTONED	Bit-map data is not halftoned. This is the default value.

BRH_ERRORDIFFUSION	Error Diffusion or Damped Error Diffusion algorithm.
BRH_PANDA	Processing Algorithm for Non-coded Document Acquisition.
BRH_SUPERCIRCLE	Super Circle algorithm.

BITMAPINFO2 Field - cSize1

cSize1 (ULONG)
Size value 1.

If BRH_ERRORDIFFUSION is specified in *usRendering* , *cSize1* is the error damping as a percentage in the range 0 through 100. A value of 100% indicates no damping, and a value of 0% indicates that any errors are not diffused.

If BRH_PANDA or BRH_SUPERCIRCLE is specified, *cSize1* is the x dimension of the pattern used, in pels.

BITMAPINFO2 Field - cSize2

cSize2 (ULONG)
Size value 2.

If BRH_ERRORDIFFUSION is specified in *usRendering* , this parameter is ignored.

If BRH_PANDA or BRH_SUPERCIRCLE is specified, *cSize2* is the y dimension of the pattern used, in pels.

BITMAPINFO2 Field - ulColorEncoding

ulColorEncoding (ULONG)
Color encoding.

BCE_RGB	Each element in the color array is an RGB2 datatype. This is the default value.

BITMAPINFO2 Field - ulIdentifier

ulIdentifier (ULONG)
Reserved for application use.

BITMAPINFO2 Field - argbColor[1]

argbColor[1] (RGB2)
Array of RGB values.

This is a packed array of 24-bit RGB values. If there are N bits per pel (N = *cPlanes* * *cBitCount*), the array contains 2**N RGB values. However, if N = 24, the bit map does not need the **color** array because the standard-format bit map, with 24 bits per pel, is assumed to contain RGB values.

BITMAPINFOHEADER

Bit-map information header structure.

Each bit plane logically contains (*cx* * *cy* * *cBitCount*) bits, although the actual length can be greater because of padding.

See also BITMAPINFOHEADER2, which is preferred.

```
typedef struct _BITMAPINFOHEADER {
  ULONG      cbFix;      /*  Length of structure. */
  USHORT     cx;         /*  Bit-map width in pels. */
  USHORT     cy;         /*  Bit-map height in pels. */
  USHORT     cPlanes;    /*  Number of bit planes. */
  USHORT     cBitCount;  /*  Number of bits per pel within a plane. */
} BITMAPINFOHEADER;

typedef BITMAPINFOHEADER *PBITMAPINFOHEADER;
```

BITMAPINFOHEADER Field - cbFix

cbFix (ULONG)
Length of structure.

BITMAPINFOHEADER Field - cx

cx (USHORT)
Bit-map width in pels.

BITMAPINFOHEADER Field - cy

cy (USHORT)
Bit-map height in pels.

BITMAPINFOHEADER Field - cPlanes

cPlanes (USHORT)
Number of bit planes.

BITMAPINFOHEADER Field - cBitCount

cBitCount (USHORT)
Number of bits per pel within a plane.

BITMAPINFOHEADER2

Bit-map information header structure.

Each bit plane logically contains (*cx* * *cy* * *cBitCount*) bits, although the actual length can be greater because of padding.

Note: Many functions can accept either this structure or the BITMAPINFOHEADER structure. Where possible, use BITMAPINFOHEADER2.

```
typedef struct _BITMAPINFOHEADER2 {
  ULONG      cbFix;            /*  Length of structure. */
  ULONG      cx;               /*  Bit-map width in pels. */
  ULONG      cy;               /*  Bit-map height in pels. */
  USHORT     cPlanes;          /*  Number of bit planes. */
  USHORT     cBitCount;        /*  Number of bits per pel within a plane. */
  ULONG      ulCompression;    /*  Compression scheme used to store the bit map. */
  ULONG      cbImage;          /*  Length of bit-map storage data, in bytes. */
  ULONG      cxResolution;     /*  Horizontal component of the resolution of target device. */
  ULONG      cyResolution;     /*  Vertical component of the resolution of target device. */
  ULONG      cclrUsed;         /*  Number of color indexes used. */
  ULONG      cclrImportant;    /*  Minimum number of color indexes for satisfactory appearance of the bit map.
  USHORT     usUnits;          /*  Units of measure. */
  USHORT     usReserved;       /*  Reserved. */
  USHORT     usRecording;      /*  Recording algorithm. */
  USHORT     usRendering;      /*  Halftoning algorithm. */
  ULONG      cSize1;           /*  Size value 1. */
  ULONG      cSize2;           /*  Size value 2. */
  ULONG      ulColorEncoding;  /*  Color encoding.

BCE_RGB   Each element in the color array is an RGB2 datatype. This is the default value.
 */
  ULONG      ulIdentifier;     /*  Reserved for application use. */
} BITMAPINFOHEADER2;

typedef BITMAPINFOHEADER2 *PBITMAPINFOHEADER2;
```

BITMAPINFOHEADER2 Field - cbFix

cbFix (ULONG)
Length of structure.

The structure can be truncated after *cBitCount* or any subsequent field.

BITMAPINFOHEADER2 Field - cx

cx (ULONG)
Bit-map width in pels.

BITMAPINFOHEADER2 Field - cy

cy (ULONG)
Bit-map height in pels.

BITMAPINFOHEADER2 Field - cPlanes

cPlanes (USHORT)
Number of bit planes.

BITMAPINFOHEADER2 Field - cBitCount

cBitCount (USHORT)
Number of bits per pel within a plane.

BITMAPINFOHEADER2 Field - ulCompression

ulCompression (ULONG)
Compression scheme used to store the bit map.

BCA_UNCOMP
Bit map is uncompressed.

BCA_HUFFMAN1D

The bit map is compressed by a modified Huffman encoding. This is valid for a bi-level (one bit per pel) bit map.

BCA_RLE4

The bit map is a 4-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

BCA_RLE8

The bit map is an 8-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

BCA_RLE24

The bit map is a 24-bit per pel run-length encoded bit map. See the following section, "Format of Compressed Data," for a description of the format of the compressed data.

Format of Compressed Data

Encoding a run length:

Run length encoded bit maps are encoded in the buffer in a controlled format. In all cases, if the first byte is non-zero, it is the length of a run of pels of a particular color or, in the case of a BCA_RLE4 bit map, a run of a length of pels of alternating colors.

```
1st-byte       pel repetition count >= 1
2nd-4th bytes  (BCA_RLE24 only) RGB value of pel.
2nd-byte       (BCA_RLE8) color index of pel to be repeated
               (BCA_RLE4) the second byte contains 2 4-bit
               color indexes.  The repetition count is
               completed by alternately choosing the high-order
               nibble followed by the low-order nibble for the
               succeeding pels until the count is exhausted.
```

Unencoded run:

An unencoded run is a string of pels to be placed in consecutive positions in the destination bit map.

```
1st-byte     0
2nd-byte     COUNT = a multiple of 3 for BCA_RLE24 bit maps, or
             COUNT >= 3 (for BCA_RLE4 and BCA_RLE8 bit maps).
```

followed by the bytes as follows:

BCA_RLE24

A string of bytes specifying the RGB color values of succeeding pels. If COUNT is odd, it must be padded by a zero byte for an even length overall.

BCA_RLE8

A string of bytes specifying color indexes for succeeding pels. If COUNT is odd, it must be padded by a zero byte for an even length overall.

BCA_RLE4

A string of bytes, each byte providing two color indexes, with the high-order nibble specifying the index of the pel preceding the low-order nibble. The COUNT specifies the number of indexes. The overall length of the string must be an even number of bytes, and thus may be padded with a zero byte, and the low order nibble of the last significant byte may also be zero and not used.

Delta record:

A delta record indicates a shift in position in the destination bit map before the next record is decoded.

```
1st-byte        0
2nd-byte        2
3rd-byte        Delta-x (unsigned)
4th-byte        Delta-y (unsigned)
```

This is a relative jump record. It implies that the next record is to be decoded into a position in the destination bit map at an offset from the current position, determined by changing the horizontal and vertical positions by Delta-x and Delta-y, respectively.

End-of-line record:

The end-of-line record signifies that the data for the current scan line is complete and that decoding of the next record should begin at the start of the next scan line.

```
1st-byte        0
2nd-byte        0
```

End-of-RLE record:

The end-of-RLE record signifies the end of the data in the RLE compressed bit map.

```
1st-byte        0
2nd-byte        1
```

BITMAPINFOHEADER2 Field - cbImage

cbImage (ULONG)
Length of bit-map storage data, in bytes.

If the bit map is uncompressed, zero (the default) can be specified for this.

BITMAPINFOHEADER2 Field - cxResolution

cxResolution (ULONG)
Horizontal component of the resolution of target device.

The resolution of the device the bit map is intended for, in the units specified by *usUnits* . This information enables applications to select from a resource group the bit map that best matches the characteristics of the current output device.

BITMAPINFOHEADER2 Field - cyResolution

cyResolution (ULONG)
Vertical component of the resolution of target device.

See the description of *cxResolution* .

BITMAPINFOHEADER2 Field - cclrUsed

cclrUsed (ULONG)
Number of color indexes used.

The number of color indexes from the color table that are used by the bit map. If this is zero (the default), all the indexes are used. If it is non-zero, only the first *cclrUsed* entries in the table are accessed by the system, and further entries can be omitted.

For the standard formats with a *cBitCount* of 1, 4, or 8 (and *cPlanes* equal to 1), any indexes beyond *cclrUsed* are invalid. For example, a bit map with 64 colors can use the 8-bitcount format without having to supply the other 192 entries in the color table. For the 24-bitcount standard format, *cclrUsed* is the number of colors used by the bit map.

BITMAPINFOHEADER2 Field - cclrImportant

cclrImportant (ULONG)
Minimum number of color indexes for satisfactory appearance of the bit map.

More colors may be used in the bit map, but it is not necessary to assign them to the device palette. These additional colors may be mapped to the nearest colors available.

Zero (the default) means that all entries are important.

For a 24-bitcount standard format bit map, the *cclrImportant* colors are also listed in the color table relating to this bit map.

BITMAPINFOHEADER2 Field - usUnits

usUnits (USHORT)
Units of measure.

Units of measure of the horizontal and vertical resolution, *cxResolution* and *cyResolution*.

BRU_METRIC	Pels per meter. This is the default value.

BITMAPINFOHEADER2 Field - usReserved

usReserved (USHORT)
Reserved.

This is a reserved field. If present, it must be zero.

BITMAPINFOHEADER2 Field - usRecording

usRecording (USHORT)
Recording algorithm.

The format in which the bit-map data is recorded.

BRA_BOTTOMUP	Scan lines are recorded bottom to top. This is the default value.

BITMAPINFOHEADER2 Field - usRendering

usRendering (USHORT)
Halftoning algorithm.

The algorithm used to record bit-map data that has been digitally halftoned.

BRH_NOTHALFTONED	Bit-map data is not halftoned. This is the default value.
BRH_ERRORDIFFUSION	Error Diffusion or Damped Error Diffusion algorithm.
BRH_PANDA	Processing Algorithm for Non-coded Document Acquisition.
BRH_SUPERCIRCLE	Super Circle algorithm.

BITMAPINFOHEADER2 Field - cSize1

cSize1 (ULONG)
Size value 1.

If BRH_ERRORDIFFUSION is specified in *usRendering* , *cSize1* is the error damping as a percentage in the range 0 through 100. A value of 100% indicates no damping, and a value of 0% indicates that any errors are not diffused.

If BRH_PANDA or BRH_SUPERCIRCLE is specified, *cSize1* is the x dimension of the pattern used, in pels.

BITMAPINFOHEADER2 Field - cSize2

cSize2 (ULONG)
Size value 2.

If BRH_ERRORDIFFUSION is specified in *usRendering* , this parameter is ignored.

If BRH_PANDA or BRH_SUPERCIRCLE is specified, *cSize2* is the y dimension of the pattern used, in pels.

BITMAPINFOHEADER2 Field - ulColorEncoding

ulColorEncoding (ULONG)
Color encoding.

BCE_RGB	Each element in the color array is an RGB2 datatype. This is the default value.

BITMAPINFOHEADER2 Field - ulIdentifier

ulIdentifier (ULONG)
Reserved for application use.

BOOL

Boolean.

Valid values are FALSE, which is 0, and TRUE, which is 1.

```
typedef unsigned long BOOL;
```

BYTE

A byte.

```
typedef unsigned char BYTE;
```

CHAR

Single-byte character.

```
#define CHAR char
```

CHARBUNDLE

Character-attributes bundle structure.

```
typedef struct _CHARBUNDLE {
  LONG       lColor;         /*  Character foreground color. */
  LONG       lBackColor;     /*  Character background color. */
  USHORT     usMixMode;      /*  Character foreground-mix mode. */
  USHORT     usBackMixMode;  /*  Character background-mix mode. */
  USHORT     usSet;          /*  Character set. */
  USHORT     usPrecision;    /*  Character precision. */
  SIZEF      sizfxCell;      /*  Character cell size. */
  POINTL     ptlAngle;       /*  Character angle. */
  POINTL     ptlShear;       /*  Character shear. */
```

```
  USHORT     usDirection;    /*  Character direction. */
  USHORT     usTextAlign;    /*  Text alignment. */
  FIXED      fxExtra;        /*  Character extra. */
  FIXED      fxBreakExtra;   /*  Character break extra. */
} CHARBUNDLE;

typedef CHARBUNDLE *PCHARBUNDLE;
```

CHARBUNDLE Field - lColor

lColor (LONG)
Character foreground color.

CHARBUNDLE Field - lBackColor

lBackColor (LONG)
Character background color.

CHARBUNDLE Field - usMixMode

usMixMode (USHORT)
Character foreground-mix mode.

CHARBUNDLE Field - usBackMixMode

usBackMixMode (USHORT)
Character background-mix mode.

CHARBUNDLE Field - usSet

usSet (USHORT)
Character set.

CHARBUNDLE Field - usPrecision

usPrecision (USHORT)
Character precision.

CHARBUNDLE Field - sizfxCell

sizfxCell (SIZEF)
Character cell size.

CHARBUNDLE Field - ptlAngle

ptlAngle (POINTL)
Character angle.

CHARBUNDLE Field - ptlShear

ptlShear (POINTL)
Character shear.

CHARBUNDLE Field - usDirection

usDirection (USHORT)
Character direction.

CHARBUNDLE Field - usTextAlign

usTextAlign (USHORT)
Text alignment.

CHARBUNDLE Field - fxExtra

fxExtra (FIXED)
Character extra.

CHARBUNDLE Field - fxBreakExtra

fxBreakExtra (FIXED)
Character break extra.

CONVCONTEXT

Dynamic-data-exchange conversation context structure.

```
typedef struct _CONVCONTEXT {
  ULONG     cb;           /*  Length of structure. */
  ULONG     fsContext;    /*  Options. */
  ULONG     idCountry;    /*  Country code. */
  ULONG     usCodepage;   /*  Code-page identity. */
  ULONG     usLangID;     /*  Language. */
  ULONG     usSubLangID;  /*  Sub-language. */
} CONVCONTEXT;

typedef CONVCONTEXT *PCONVCONTEXT;
```

CONVCONTEXT Field - cb

cb (ULONG)
Length of structure.

This must be set to the length of the CONVCONTEXT structure.

CONVCONTEXT Field - fsContext

fsContext (ULONG)
Options.

DDECTXT_CASESENSITIVE
All strings in this conversation are case sensitive.

CONVCONTEXT Field - idCountry

idCountry (ULONG)
Country code.

CONVCONTEXT Field - usCodepage

usCodepage (ULONG)
Code-page identity.

CONVCONTEXT Field - usLangID

usLangID (ULONG)
Language.

Zero is valid and means no language information.

CONVCONTEXT Field - usSubLangID

usSubLangID (ULONG)
Sub-language.

Zero is valid and means no sub-language information.

DDEINIT

Dynamic-data-exchange initiation structure.

```
typedef struct _DDEINIT {
  ULONG     cb;              /*  Length of structure. */
  PSZ       pszAppName;      /*  Application name. */
  PSZ       pszTopic;        /*  Topic. */
  ULONG     offConvContext;  /*  Conversation context. */
} DDEINIT;

typedef DDEINIT *PDDEINIT;
```

DDEINIT Field - cb

cb (ULONG)
Length of structure.

This must be set to the length of the DDEINIT structure.

DDEINIT Field - pszAppName

pszAppName (PSZ)
Application name.

Pointer to name of the server application.

Application names must not contain slashes or backslashes. These characters are reserved for future use in network implementations.

DDEINIT Field - pszTopic

pszTopic (PSZ)
Topic.

Pointer to name of the topic.

DDEINIT Field - offConvContext

offConvContext (ULONG)
Conversation context.

Offset to a CONVCONTEXT structure.

DDESTRUCT

Dynamic-data-exchange control structure.

```
typedef struct _DDESTRUCT {
  ULONG      cbData;         /*  Length of the data. */
```

```
  USHORT     fsStatus;       /*  Status of the data exchange. */
  USHORT     usFormat;       /*  DDE data format. */
  USHORT     offszItemName;  /*  Offset to item name. */
  USHORT     offabData;      /*  Offset to beginning of data. */
} DDESTRUCT;

typedef DDESTRUCT *PDDESTRUCT;
```

DDE formats, specified in the *usFormat* parameter, are registered with the atom manager, using the system atom table. The predefined DDE formats are guaranteed not to conflict with the values returned by the atom manager.

Applications can define their own data formats; however, each nonstandard DDE format must have a unique identification number. To receive an identification number for a nonstardard format, the application must register the name of the format in the system atom table. Other applications that have the name of the format can then query the system atom table for the format's identification number. This method ensures that all applications use the same atom to identify a format.

DDESTRUCT Field - cbData

cbData (ULONG)
Length of the data.

This is the length of data that occurs after the *offabData* parameter. If no data exists, this field should contain a zero (0).

DDESTRUCT Field - fsStatus

fsStatus (USHORT)
Status of the data exchange.

DDE_FACK
Positive acknowledgement
DDE_FBUSY
Application is busy
DDE_FNODATA
No data transfer for advise
DDE_FACKREQ
Acknowledgements are requested
DDE_FRESPONSE
Response to WM_DDE_REQUEST
DDE_NOTPROCESSED
DDE message not understood
DDE_FAPPSTATUS
A 1-byte field of bits that are reserved for application-specific returns.

DDESTRUCT Field - usFormat

usFormat (USHORT)
DDE data format.

This parameter can be set to one of the following values:

DDEFMT_TEXT
: System-defined standard format for exchanging text data.

SZFMT_BITMAP
: Specifies that the data is a bit map.

SZFMT_CPTEXT
: Specifies text whose format is defined by a CPTEXT data structure. Applications can use this format to pass multiple-language strings without changing the conversation context.

SZFMT_DIF
: Specifies that the data is in Data Image Format (DIF).

SZFMT_DSPBITMAP
: Specifies that the data is a bit-map representation of a private data format.

SZFMT_DSPMETAFILE
: Specifies that the data is a metafile representation of a private data format.

SZFMT_DSPMETAFILEPICT
: Specifies that the data is a metafile picture representation of a private data format.

SZFMT_DSPTEXT
: Specifies that the data is a text representation of a private data format.

SZFMT_LINK
: Specifies that the data is in link-file format.

SZFMT_METAFILE
: Specifies that the data is a metafile.

SZFMT_METAFILEPICT
: Specifies that the data is a metafile picture defined by an MFP data structure.

SZFMT_OEMTEXT
: Specifies that the data is in OEM Text format.

SZFMT_PALETTE
: Specifies that the data is in palette format.

SZFMT_SYLK
: Specifies that the data is in Synchronous Link format.

SZFMT_TEXT
: Specifies that the data is an array of text characters. These characters can include new-line characters to indicate linebreaks. The zero-length character indicates the end of the text data.

SZFMT_TIFF
: Specifies that the data is in Tag Image File Format (TIFF).

DDESTRUCT Field - offszItemName

offszItemName (USHORT)
: Offset to item name.

This is the offset to the item name from the start of this structure. Item name is a null (0x00) terminated string. If no item name exists, there must be a single null (0x00) character in this position. (That is, ItemName is ALWAYS a null terminated string.)

DDESTRUCT Field - offabData

offabData (USHORT)
: Offset to beginning of data.

This is the offset to the data, from the start of this structure. This field should be calculated regardless of the presence of data. If no data exists, **cbData** must be zero (0).

For compatibility reasons, this data should not contain embedded pointers. Offsets should be used instead.

DEVOPENSTRUC

Open-device data structure.

```
typedef struct _DEVOPENSTRUC {
  PSZ           pszLogAddress;       /*  Logical address. */
  PSZ           pszDriverName;       /*  Driver name. */
  PDRIVDATA     pdriv;               /*  Driver data. */
  PSZ           pszDataType;         /*  Data type. */
  PSZ           pszComment;          /*  Comment. */
  PSZ           pszQueueProcName;    /*  Queue-processor name. */
  PSZ           pszQueueProcParams;  /*  Queue-processor parameters. */
  PSZ           pszSpoolerParams;    /*  Spooler parameters. */
  PSZ           pszNetworkParams;    /*  Network parameters. */
} DEVOPENSTRUC;

typedef DEVOPENSTRUC *PDEVOPENSTRUC;
```

DEVOPENSTRUC Field - pszLogAddress

pszLogAddress (PSZ)
Logical address.

This is required for an OD_DIRECT device being opened with DevOpenDC; it is the logical device address, such as "LPT1" on OS/2. Some drivers may accept a file name for this parameter or even a named pipe.

Where output is to be queued (for an OD_QUEUED device), this is the name of the queue for the output device. The queue name can be a UNC name.

Note: This parameter can be a port name for a printer device context.

DEVOPENSTRUC Field - pszDriverName

pszDriverName (PSZ)
Driver name.

Character string identifying the printer driver, for example, LASERJET. The Default Device Driver field of the PRQINFO3 structure, associated with the required print queue, gives the driver and device name, separated by a period, for example LASERJET.HP LaserJet IIID. It can contain only the name up to the period, for example LASERJET.

DEVOPENSTRUC Field - pdriv

pdriv (PDRIVDATA)
Driver data.

Data that is to be passed directly to the PM device driver. Whether any of this is required depends upon the device driver.

For printer device context, this is a pointer to the job properties data.

DEVOPENSTRUC Field - pszDataType

pszDataType (PSZ)
Data type.

For an OD_QUEUED or OD_DIRECT device, this parameter defines the type of data that is to be queued as follows:

PM_Q_STD	Standard format
PM_Q_RAW	Raw format

Note that a device driver can define other data types.

For OD_QUEUED or OD_DIRECT defice types, the default is supplied by the device driver if *pszDataType* is not specified. For any other device type, *pszDataType* is ignored.

DEVOPENSTRUC Field - pszComment

pszComment (PSZ)
Comment.

Optional character string that the printer object displays to the user in a job settings notebook. It is recommended that the application include its own name in this comment string.

Note: The job title text is derived from the document name passed to DevEscape (DEVESC_STARTDOC).

DEVOPENSTRUC Field - pszQueueProcName

pszQueueProcName (PSZ)
Queue-processor name.

This is the name of the queue processor, for queued output, and is usually the default.

DEVOPENSTRUC Field - pszQueueProcParams

pszQueueProcParams (PSZ)
Queue-processor parameters.

Queue processor parameters (optional). They can include information such as the number of copies you want to print and the size of the output area on the printed page.

The first parameter (*COP*) is used for all spool-file formats. The remaining parameters are valid for PM_Q_STD spool files only. Because PM_Q_STD data are used mainly for *graphic* data, these parameters are described in relation to the printing of picture files.

The PMPRINT/PMPLOT queue-processor parameters are separated by spaces and are:

COP=**n**

The *COP* parameter specifies the number of copies of the spool file that you want printed. The value of *n* must be an integer in the range of 1 through 999.

The default is *COP=1* .

ARE=C | **w,h,l,t**

The *ARE* parameter determines the size and position of the output area. This is the area of the physical page to which printing is restricted.

The default value of *ARE=C* means that the output area is the whole page. Note, however, that the printer cannot print outside its own device clip limits.

To size and position the output area at a specific point on the page, use *ARE=* **w,h,l,t**, where:

w, h	are the width and height of the desired output area.
l, t	are the offsets of the upper-left corner of the output area from the left (l) and from the top (t) of the maximum output area.

These four values must be given as percentages of the maximum output dimensions. The maximum output area is the area within the device clip limits.

FIT=S | **l,t**

The *FIT* parameter determines which part of the picture is to be printed. You can request the whole of the picture, scaled to fit the output area; or you can position the picture (actual size) anywhere within the output area. This could mean that the picture is clipped at the boundaries of the output area.

The default value of *FIT=S* causes the output to be scaled until the larger of the height or width just fits within the defined output area. The aspect ratio of the picture is maintained.

To print the picture in actual size, use *FIT=* **l,t**, where *l,t* are the coordinates of the point in the picture that you want positioned at the center of the output area: *l* is measured from the left edge of the picture; and *t* is measured from the top edge. The coordinates must be given as percentages of the actual dimensions of the picture.

XFM=0 | 1

The *XFM* parameter enables you to override the picture-positioning and clipping instructions that are provided by the *ARE* and *FIT* parameters, including their defaults.

The default value of *XFM=1* allows the appearance of the output to be determined by the settings of the *ARE* and *FIT* parameters.

A value of *XFM=0* yields output as specified in the picture file. For example, applications that use many different forms can define different positions on each form for their output.

COL=M | C

The *COL* parameter enables you to specify color output if you have a color printer.

A value of *COL=M* creates monochrome output (black foreground with no background color). This is supported by all devices.

A value of *COL=C* creates color output. If you request color output on a monochrome device, the printer presentation driver tries to satisfy your request, which can cause problems because the only color available is black. For example, if the picture file specifies a red line on a blue background, both are drawn in black.

The default is *COL=M* when you are addressing a monochrome printer and *COL=C* when you are addressing a color printer.

MAP=N | A

The *MAP* parameter enables you to decide how the *neutral* colors (those that are not specified in the picture file) are printed.

The default value of *MAP=N* yields a *normal* representation of the screen picture on a printed page, which means that the page background is white and the foreground is black.

A value of *MAP=A* provides the reverse of the normal representation: the background is black and the foreground is white on the printed page.

CDP=codepage

The *CDP* parameter overrides the codepage to being used for PM_Q_RAW print jobs. The print queue driver uses DEVESC_SETMODE to set the codepage, but not all printer drivers support this device escape.

XLT=0 | 1

The *XLT* parameter can eliminate the translation component when printing a metafile if *XLT=1* .

When the resolution of the device is higher than that of the world coordinate space, a small translation of world coordinate point (0,0) occurs on the device to preserve the accuracy of the mapping from world to device coordinate units. For example, (0,0) becomes (1,1) if there are 3 pels to every world coordinate.

Normally, this is not noticeable, but it can be a problem with some devices. For example, in order to draw a complete row of 80 characters using a device font, a device may require the text to start at device coordinate position zero. Starting at a position other than zero may cause one or more characters at the end of the row to be clipped. In such cases, elimination of the translation is important and can be accomplished by specifying *XLT=1* .

The default is *XLT=0* .

DEVOPENSTRUC Field - pszSpoolerParams

pszSpoolerParams (PSZ)
Spooler parameters.

Spooler parameters (optional) are separated by spaces. They are used for scheduling print jobs and are as follows:

- The form names that identify the paper to be used, for example, FORM=A4,A5,ENV. The form names are optional; but if they are provided, the spooler is able to hold off printing the jobs until the required form is installed in the printer. If the form name is not provided, the spooler attempts to print the job. The printer driver recognizes that there is a forms problem and displays a FORMS MISMATCH message box.
- Priority of the print job, for example, *PRTY=60* . The priority is specified as an integer in the range 1 through 99; 99 is the highest. The default priority value is 50. The application can use the spooler priority parameter to prioritize its own jobs; however, it is not good practice for an application always to use priority 99 in an attempt to get its jobs printed first.

DEVOPENSTRUC Field - pszNetworkParams

pszNetworkParams (PSZ)
Network parameters.

Optional parameter that can be used to specify network options; for example, *USER=JOESMITH* .

DRIVDATA

Driver-data structure.

```
typedef struct _DRIVDATA {
  LONG     cb;               /*  Length. */
  LONG     lVersion;         /*  Version. */
  CHAR     szDeviceName[32]; /*  Device name. */
  CHAR     abGeneralData[1]; /*  General data. */
} DRIVDATA;

typedef DRIVDATA *PDRIVDATA;
```

DRIVDATA Field - cb

cb (LONG)
Length.

The length of the structure.

DRIVDATA Field - lVersion

lVersion (LONG)
Version.

The version number of the data. Version numbers are defined by particular PM device drivers.

DRIVDATA Field - szDeviceName[32]

szDeviceName[32] (CHAR)
Device name.

A string in a 32-byte field identifying the particular device (model number, and so on). Again, valid values are defined by PM device drivers.

DRIVDATA Field - abGeneralData[1]

abGeneralData[1] (CHAR)
General data.

Data as defined by the Presentation Manager device driver.

The data type of this field is defined by the Presentation Manager device driver. It does not contain pointers, as these are not necessarily valid when passed to the device driver.

ERRORID

Error identity.

```
typedef ULONG ERRORID;
```

ERRINFO

Error-information structure.

```
typedef struct _ERRINFO {
  ULONG       cbFixedErrInfo;  /*  Length of fixed data to this structure. */
  ERRORID     idError;         /*  Error identity. */
  ULONG       cDetailLevel;    /*  Number of levels of detail. */
  ULONG       offaoffszMsg;    /*  Offset to the array of message offsets. */
  ULONG       offBinaryData;   /*  Offset to the binary data. */
} ERRINFO;

typedef ERRINFO *PERRINFO;
```

ERRINFO Field - cbFixedErrInfo

cbFixedErrInfo (ULONG)
Length of fixed data to this structure.

ERRINFO Field - idError

idError (ERRORID)
Error identity.

This is identical to the value returned by WinGetLastError.

ERRINFO Field - cDetailLevel

cDetailLevel (ULONG)
Number of levels of detail.

This is the number of entries in the array of words pointed to by the following field. One level of detail is provided.

ERRINFO Field - offaoffszMsg

offaoffszMsg (ULONG)

Offset to the array of message offsets.

This is an offset to an array of 16-bit offsets to null-terminated strings. Each string is a printable message that offers varying levels of information. The first level is the least amount of detail, and the remaining levels offer more and more detail.

The first level of detail is always an error message string, in the following format:

```
xxxnnnns

where    xxx    is the product identifier
         nnnn   is the message number
         s      is the message severity letter
                  W = warning
                  E = error
                  S = severe error
                  U = unrecoverable
```

ERRINFO Field - offBinaryData

offBinaryData (ULONG)
Offset to the binary data.

This can contain additional information relating to the error.

FACENAMEDESC

Face-name description structure. See GpiQueryFaceString.

```
typedef struct _FACENAMEDESC {
  USHORT     usSize;         /*  Length of structure. */
  USHORT     usWeightClass;  /*  Weight class. */
  USHORT     usWidthClass;   /*  Width class. */
  USHORT     usReserved;     /*  Reserved. */
  ULONG      flOptions;      /*  Other characteristics of the font. */
} FACENAMEDESC;

typedef FACENAMEDESC *PFACENAMEDESC;
```

FACENAMEDESC Field - usSize

usSize (USHORT)
Length of structure.

FACENAMEDESC Field - usWeightClass

usWeightClass (USHORT)
Weight class.

Indicates the visual weight (thickness of strokes) of the characters in the font:

FWEIGHT_DONT_CARE	Any font weight satisfies the request.
FWEIGHT_ULTRA_LIGHT	Ultra-light.
FWEIGHT_EXTRA_LIGHT	Extra-light.
FWEIGHT_LIGHT	Light.
FWEIGHT_SEMI_LIGHT	Semi-light.
FWEIGHT_NORMAL	Medium (normal) weight.
FWEIGHT_SEMI_BOLD	Semi-bold.
FWEIGHT_BOLD	Bold.
FWEIGHT_EXTRA_BOLD	Extra-bold.
FWEIGHT_ULTRA_BOLD	Ultra-bold.

FACENAMEDESC Field - usWidthClass

usWidthClass (USHORT)
Width class.

Indicates the relative aspect ratio of the characters of the font in relation to the normal aspect ratio for this type of font:

FWIDTH_DONT_CARE	Any font width satisfies the request.
FWIDTH_ULTRA_CONDENSED	Ultra-condensed (50% of normal).
FWIDTH_EXTRA_CONDENSED	Extra-condensed (62.5% of normal).
FWIDTH_CONDENSED	Condensed (75% of normal).
FWIDTH_SEMI_CONDENSED	Semi-condensed (87.5% of normal).
FWIDTH_NORMAL	Medium (normal).
FWIDTH_SEMI_EXPANDED	Semi-expanded (112.5% of normal).
FWIDTH_EXPANDED	Expanded (125% of normal).
FWIDTH_EXTRA_EXPANDED	Extra-expanded (150% of normal).
FWIDTH_ULTRA_EXPANDED	Ultra-expanded (200% of normal).

FACENAMEDESC Field - usReserved

usReserved (USHORT)
Reserved.

FACENAMEDESC Field - flOptions

flOptions (ULONG)
Other characteristics of the font.

FTYPE_ITALIC
Italic font required. If not specified, non-italic font required.

FTYPE_ITALIC_DONT_CARE
Italic and non-italic fonts can satisfy the request. If this option is specified, FTYPE_ITALIC is ignored.

FTYPE_OBLIQUE
Oblique font required. If not specified, non-oblique font required.

FTYPE_OBLIQUE_DONT_CARE
Oblique and non-oblique fonts can satisfy the request. If this option is specified, FTYPE_OBLIQUE is ignored.

FTYPE_ROUNDED
Rounded font required. If not specified, non-rounded font required.

FTYPE_ROUNDED_DONT_CARE
Rounded and non-rounded fonts can satisfy the request. If this option is specified, FTYPE_ROUNDED is ignored.

FATTRS

Font-attributes structure.

```
typedef struct _FATTRS {
  USHORT     usRecordLength;        /*  Length of record. */
  USHORT     fsSelection;           /*  Selection indicators. */
  LONG       lMatch;                /*  Matched-font identity. */
  CHAR       szFacename[FACESIZE];  /*  Typeface name. */
  USHORT     idRegistry;            /*  Registry identifier. */
  USHORT     usCodePage;            /*  Code page. */
  LONG       lMaxBaselineExt;       /*  Maximum baseline extension. */
  LONG       lAveCharWidth;         /*  Average character width. */
  USHORT     fsType;                /*  Type indicators. */
  USHORT     fsFontUse;             /*  Font-use indicators. */
} FATTRS;

typedef FATTRS *PFATTRS;
```

FATTRS Field - usRecordLength

usRecordLength (USHORT)
Length of record.

FATTRS Field - fsSelection

fsSelection (USHORT)
Selection indicators.

Flags causing the following features to be simulated by the system.

Note: If an italic flag is applied to a font that is itself defined as italic, the font is slanted further by italic simulation.

Underscore or strikeout lines are drawn using the appropriate attributes (for example, color) from the character bundle (see the CHARBUNDLE datatype), not the line bundle (see LINEBUNDLE). The width of the line, and the vertical position of the line in font space, are determined by the font. Horizontally, the line starts from a point in font space directly above or below the start point of each character, and extends to a point directly above or below the escapement point for that character.

For this purpose, the start and escapement points are those applicable to left-to-right or right-to-left character directions (see GpiSetCharDirection in *Graphics Programming Interface Programming Reference*), even if the string is currently being drawn in a top-to-bottom or bottom-to-top direction.

For left-to-right or right-to-left directions, any white space generated by the character extra and character break extra attributes (see GpiSetCharExtra and GpiSetCharBreakExtra in *Graphics Programming Interface Programming Reference*), as well as increments provided by the vector of increments on GpiCharStringPos and GpiCharStringPosAt, are also underlined/overstruck, so that in these cases the line is continuous for the string.

FATTR_SEL_ITALIC	Generate *italic* font.
FATTR_SEL_UNDERSCORE	Generate underscored font.
FATTR_SEL_BOLD	Generate **bold** font. (Note that the resulting characters are wider than those in the original font.)
FATTR_SEL_STRIKEOUT	Generate font with overstruck characters.
FATTR_SEL_OUTLINE	Use an outline font with hollow characters. If this flag is not set, outline font characters are filled. Setting this flag normally gives better performance, and for sufficiently small characters (depending on device resolution) there may be little visual difference.

FATTRS Field - lMatch

lMatch (LONG)
Matched-font identity.

FATTRS Field - szFacename[FACESIZE]

szFacename[FACESIZE] (CHAR)
Typeface name.

The typeface name of the font, for example, Tms Rmn.

FATTRS Field - idRegistry

idRegistry (USHORT)
Registry identifier.

Font registry identifier (zero if unknown).

FATTRS Field - usCodePage

usCodePage (USHORT)
Code page.

If zero, the current Gpi code page (see GpiSetCp in *Graphics Programming Interface Programming Reference*) is used. A subsequent GpiSetCp function changes the code page used for this logical font.

FATTRS Field - lMaxBaselineExt

lMaxBaselineExt (LONG)
Maximum baseline extension.

For raster fonts, this should be the height of the required font, in world coordinates.

For outline fonts, this should be zero.

FATTRS Field - lAveCharWidth

lAveCharWidth (LONG)
Average character width.

For raster fonts, this should be the width of the required font, in world coordinates.

For outline fonts, this should be zero.

FATTRS Field - fsType

fsType (USHORT)
Type indicators.

FATTR_TYPE_KERNING	Enable kerning (PostScript only).
FATTR_TYPE_MBCS	Font for mixed single- and double-byte code pages.
FATTR_TYPE_DBCS	Font for double-byte code pages.
FATTR_TYPE_ANTIALIASED	Antialiased font required. Only valid if supported by the device driver.

FATTRS Field - fsFontUse

fsFontUse (USHORT)
Font-use indicators.

These flags indicate how the font is to be used. They affect presentation speed and font quality.

FATTR_FONTUSE_NOMIX	Text is not mixed with graphics and can be written without regard to any interaction with graphics objects.
FATTR_FONTUSE_OUTLINE	Select an outline (vector) font. The font characters can be used as part of a path definition. If this flag is not set, an outline font might or might not be selected. If an outline font is selected, however, character widths are rounded to an integral number of pels.
FATTR_FONTUSE_TRANSFORMABLE	Characters can be transformed (for example, scaled, rotated, or sheared).

FIXED

Signed-integer fraction (16:16). This can be treated as a LONG where the value has been multiplied by 65 536.

```
typedef LONG FIXED;
```

FFDESCS

Font-file descriptor.

```
typedef CHAR FFDESCS[2][FACESIZE];
```

FFDESCS2

Font-file descriptor.

```
typedef struct _FFDESCS2 {
  ULONG     cbLength;          /*  Structure length. */
  ULONG     cbFacenameOffset;  /*  Offset of Facename in the structure. */
  BYTE      abFamilyName[1];   /*  Family name. */
} FFDESCS2;

typedef FFDESCS2 *PFFDESC2;
```

FFDESCS2 Field - cbLength

cbLength (ULONG)
Structure length.

cbLength is the overall length of the FFDESCS2 structure. It is always rounded up to a multiple of four.

FFDESCS2 Field - cbFacenameOffset

cbFacenameOffset (ULONG)
Offset of Facename in the structure.

The facename is a null terminated string. It starts at *cbFacenameOffset* bytes offset into FFDESCS2.

FFDESCS2 Field - abFamilyName[1]

abFamilyName[1] (BYTE)
Family name.

abFamilyName is a null terminated string.

FOCAMETRICS

FOCAMETRICS data structure.

This structure is returned to applications on the GPIQueryFonts and GPIQueryFontMetrics calls and conveys information from the font creator to the application.

```
typedef struct _FOCAMETRICS {
  ULONG      ulIdentity;                    /*  Structure identity code. */
  ULONG      ulSize;                        /*  Structure size in bytes. */
  CHAR       szFamilyname[32];  /*  Font family name. */
```

```
  CHAR       szFacename[32];    /*  Face name. The typeface name defines the particular font; for example, "Tim
  SHORT      usRegistryId;                       /*  Registry identifier. */
  USHORT     usCodePage;                         /*  Code page supported by the font. */
  SHORT      yEmHeight;                          /*  Em height. */
  SHORT      yXHeight;                           /*  X height. */
  SHORT      yMaxAscender;                       /*  Maximum ascender. */
  SHORT      yMaxDescender;                      /*  Maximum descender. */
  SHORT      yLowerCaseAscent;                   /*  Lowercase ascent. */
  SHORT      yLowerCaseDescent;                  /*  Lowercase descent. */
  SHORT      yInternalLeading;                   /*  Internal leading. */
  SHORT      yExternalLeading;                   /*  External leading. */
  SHORT      xAveCharWidth;                      /*  Average character width. */
  SHORT      xMaxCharInc;                        /*  Maximum character increment. */
  SHORT      xEmInc;                             /*  Em increment. */
  SHORT      yMaxBaselineExt;                    /*  Maximum baseline extent. */
  SHORT      sCharSlope;                         /*  Character slope. */
  SHORT      sInlineDir;                         /*  Inline direction. */
  SHORT      sCharRot;                           /*  Character rotation. */
  USHORT     usWeightClass;                      /*  Weight class. */
  USHORT     usWidthClass;                       /*  Width class. */
  SHORT      xDeviceRes;                         /*  X-device resolution. */
  SHORT      yDeviceRes;                         /*  Y-device resolution. */
  SHORT      usFirstChar;                        /*  First character. */
  SHORT      usLastChar;                         /*  Last character. */
  SHORT      usDefaultChar;                      /*  Default character. */
  SHORT      usBreakChar;                        /*  Break character. */
  SHORT      usNominalPointSize;                 /*  Nominal point size. */
  SHORT      usMinimumPointSize;                 /*  Minimum point size. */
  SHORT      usMaximumPointSize;                 /*  Maximum point size. */
  SHORT      usTypeFlags;                        /*  Type indicators. */
  SHORT      fsDefn;                             /*  Definition indicators. */
  SHORT      fsSelectionFlags;                   /*  Selection indicators. */
  SHORT      fsCapabilities;                     /*  Capabilities. */
  SHORT      ySubscriptXSize;                    /*  Subscript X-size. */
  SHORT      ySubscriptYSize;                    /*  Subscript Y-size. */
  SHORT      ySubscriptXOffset;                  /*  Subscript X-offset. */
  SHORT      ySubscriptYOffset;                  /*  Subscript Y-offset. */
  SHORT      ySuperscriptXSize;                  /*  Superscript X-size. */
  SHORT      ySuperscriptYSize;                  /*  Superscript Y-size. */
  SHORT      ySuperscriptXOffset;                /*  Superscript X-offset. */
  SHORT      ySuperscriptYOffset;                /*  Superscript Y-offset. */
  SHORT      yUnderscoreSize;                    /*  Underscore size. */
  SHORT      yUnderscorePosition;                /*  Underscore position. */
  SHORT      yStrikeoutSize;                     /*  Strikeout size. */
  SHORT      yStrikeoutPosition;                 /*  Strikeout position. */
  SHORT      usKerningPairs;                     /*  Kerning pairs. */
  SHORT      sFamilyClass;                       /*  Font family design classification. */
  PSZ        pszDeviceNameOffset;                /*  Address where device name is stored. */
} FOCAMETRICS;

typedef FOCAMETRICS *PFOCAMETRICS;
```

FOCAMETRICS Field - ulIdentity

ulIdentity (ULONG)
Structure identity code.

Value must be one (1). This field is not part of the FONTMETRICS structure.

FOCAMETRICS Field - ulSize

ulSize (ULONG)
Structure size in bytes.

This must be set to the size of the FOCAMETRICS structure. This field is not part of the FONTMETRICS structure.

FOCAMETRICS Field - szFamilyname[32]

szFamilyname[32] (CHAR)
Font family name.

The family name defines the basic appearance of the font; for example, "Times New Roman."** This string is null terminated and is, therefore, limited to 32 characters in length.

FOCAMETRICS Field - szFacename[32]

szFacename[32] (CHAR)
Face name. The typeface name defines the particular font; for example, "Times New Roman Bold Italic."** This string is null terminated and is, therefore, limited to 32 characters in length.

FOCAMETRICS Field - usRegistryId

usRegistryId (SHORT)
Registry identifier.

The IBM registered number (or zero).

FOCAMETRICS Field - usCodePage

usCodePage (USHORT)
Code page supported by the font.

Defines the registered code page supported by the font. A value of zero (0) implies that the font may be used with any of the code pages supported by OS/2.

When a font contains special symbols for which there is no registered code page, then code page 65400 is used.

FOCAMETRICS Field - yEmHeight

yEmHeight (SHORT)
Em height.

The height of the Em square in world coordinate units. This height corresponds to the point size for the font.

FOCAMETRICS Field - yXHeight

yXHeight (SHORT)
X height.

The nominal height above the baseline for lowercase characters (ignoring ascenders), in world coordinate units.

FOCAMETRICS Field - yMaxAscender

yMaxAscender (SHORT)
Maximum ascender.

The maximum height above the baseline reached by any part of any symbol in the font, in world coordinate units. This field may exceed *yEmHeight* .

FOCAMETRICS Field - yMaxDescender

yMaxDescender (SHORT)
Maximum descender.

The maximum depth below the baseline reached by any part of any symbol in the font, in world coordinate units. This field may exceed *yEmHeight* .

FOCAMETRICS Field - yLowerCaseAscent

yLowerCaseAscent (SHORT)
Lowercase ascent.

The maximum height above the baseline reached by any part of any lowercase (Latin unaccented ' a ' through ' z ') symbol in the font, in world coordinate units.

FOCAMETRICS Field - yLowerCaseDescent

yLowerCaseDescent (SHORT)
Lowercase descent.

The maximum depth below the baseline reached by any part of any lowercase (Latin unaccented ' a ' through ' z ') symbol in the font, in world coordinate units.

FOCAMETRICS Field - yInternalLeading

yInternalLeading (SHORT)
Internal leading.

The amount of space which, when subtracted from yMaxAscender, gives a measure of the distance above the baseline that characters extend. This measure is font-design dependent, but glyph-set independent. Therefore, this calculation approximates the visual top to a row of characters without actually looking at the characters in the row.

It is recommended that applications use this field to position the first line of a block of text by subtracting yInternalLeading from yMaxAscender. Then position the baseline that distance below the preceding information (text or graphic element).

Note: This use does not guarantee that characters will not overwrite those above them; it does give a font designer's view of where to place the text. Collision should be tested for and additional space allocated if necessary.

FOCAMETRICS Field - yExternalLeading

yExternalLeading (SHORT)
External leading.

The amount of guaranteed white space advised by the font designer to appear between adjacent rows of text. This value may be zero (0).

Note: The fonts built into Presentation Manager have zero (0) in this field.

FOCAMETRICS Field - xAveCharWidth

xAveCharWidth (SHORT)
Average character width.

This is determined by multiplying the width of each lowercase character by a constant, adding the products, and then dividing by 1000. The letters, plus their constants, are listed below:

Letter	Constant
a	64
b	14
c	27
d	35
e	100
f	20
g	14
h	42

i	63
j	3
k	6
l	35
m	20
n	56
o	56
p	17
q	4
r	49
s	56
t	71
u	31
v	10
w	18
x	3
y	18
z	2
space	166

Note: For fixed pitch fonts, this value will be the same as the (A width + B width + C width) escapement of each character.

FOCAMETRICS Field - xMaxCharInc

xMaxCharInc (SHORT)
Maximum character increment.

The maximum character increment for the font, in world coordinate units.

FOCAMETRICS Field - xEmInc

xEmInc (SHORT)
Em increment.

The width of the Em square in world coordinate units. This width corresponds to the point size of the font. When the horizontal device resolution equals the vertical device resolution, width is equal to the em height.

FOCAMETRICS Field - yMaxBaselineExt

yMaxBaselineExt (SHORT)
Maximum baseline extent.

The maximum vertical space occupied by the font, in world coordinate units. This space is the sum of *yMaxAscender* and *yMaxDescender* if both are positive. It is also the sum of *yInternalLeading* and *yEmHeight*.

One possible line spacing can be computed by adding *yMaxBaselineExt* to *yInternalLeading*. Such a line spacing, however, would be dependent on the glyph set included in the font. If a new version of the font is made available with new glyphs, then it is possible that this value will change because one of the new glyphs has gone above the previous *yMaxAscender* or below the previous *yMaxDescender*. More sophisticated applications will base their line spacing on the point size (*yEmHeight*) of the font, which is an invariant of the font, multiplied by some factor (for example, 120%), plus any external leading.

yMaxBaselineExt may exceed *yEmHeight* .

FOCAMETRICS Field - sCharSlope

sCharSlope (SHORT)
Character slope.

Defines the nominal slope of the characters of a font. The slope is defined in degrees, increasing clockwise from the vertical. An *italic* font is an example of a font with a nonzero slope.

Note: The units for this metric are degrees and minutes, encoded as shown in the following example.

```
180 degrees 59 minutes would be represented as:

    < byte 1  >     < byte 2   >

    < Minutes >     < Degrees >

 0 1 1 1 0 1 1   0 1 0 1 1 0 1 0 0

     59 min         180 degrees
```

FOCAMETRICS Field - sInlineDir

sInlineDir (SHORT)
Inline direction.

The direction in which the characters in the font are designed for viewing. Shown in degrees, increasing clockwise from the horizontal (left-to-right). Characters are added to a line of text in the inline direction.

Note: The units for this metric are degrees and minutes, encoded as shown in *sCharSlope* .

FOCAMETRICS Field - sCharRot

sCharRot (SHORT)
Character rotation.

The rotation of the character glyphs with respect to the baseline, the angle increasing counterclockwise. This is the angle assigned by the font designer.

Note: The units for this metric are degrees and minutes, encoded as shown in *sCharSlope* .

FOCAMETRICS Field - usWeightClass

usWeightClass (USHORT)
Weight class.

Indicates the visual weight (thickness of strokes) of the characters in the font:

Value	Description
1000	Ultra-light
2000	Extra-light
3000	Light
4000	Semi-light
5000	Medium (normal)
6000	Semi-bold
7000	Bold
8000	Extra-bold
9000	Ultra-bold

FOCAMETRICS Field - usWidthClass

usWidthClass (USHORT)
Width class.

Indicates the relative aspect ratio of the font characters in relation to the normal aspect ratio for this type of font:

Value	Description	% of Normal Width
1000	Ultra-condensed	50
2000	Extra-condensed	62.5
3000	Condensed	75
4000	Semi-condensed	87.5
5000	Medium (normal)	100
6000	Semi-expanded	112.5
7000	Expanded	125
8000	Extra-expanded	150
9000	Ultra-expanded	200

FOCAMETRICS Field - xDeviceRes

xDeviceRes (SHORT)
X-device resolution.

For bit-map fonts, this is the resolution in the X direction of the intended target device, measured in pels per inch.

For outline fonts, this is the number of notional units in the X direction of the Em square, measured in notional units per Em. (Notional units are the units in which the outline is defined.)

FOCAMETRICS Field - yDeviceRes

yDeviceRes (SHORT)
Y-device resolution.

For bit-map fonts, this is the resolution in the Y direction of the intended target device, measured in pels per inch.

For outline fonts, this is the number of notional units in the Y direction of the Em square, measured in notional units per Em. (Notional units are the units in which the outline is defined.)

FOCAMETRICS Field - usFirstChar

usFirstChar (SHORT)
First character.

The code point of the first character in the font.

FOCAMETRICS Field - usLastChar

usLastChar (SHORT)
Last character.

The code point of the last character in the font, expressed as an offset from *usFirstChar* .

All code points between the first and last character specified must be supported by the font.

FOCAMETRICS Field - usDefaultChar

usDefaultChar (SHORT)
Default character.

The code point that is used if a code point outside the range supported by the font is used, expressed as an offset from *usFirstChar* .

FOCAMETRICS Field - usBreakChar

usBreakChar (SHORT)
Break character.

The code point that represents the "space" or "break" character for this font, expressed as an offset from *usFirstChar* . For example, if the first character is the space in code page 850, *usFirstChar* = 32, and *usBreakChar* = 0.

FOCAMETRICS Field - usNominalPointSize

usNominalPointSize (SHORT)
Nominal point size.

For a bit-map font, this field contains the height of the font.

For an outline font, this field contains the height that the font designer intended for this font. For example, some fonts are designed for text use, in which case a value of 120 (12 point) would probably be placed in this field. Other fonts are designed for "display" use ("display" is a typographer's term for larger point sizes). This is not the only size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

FOCAMETRICS Field - usMinimumPointSize

usMinimumPointSize (SHORT)
Minimum point size.

For a bit-map font, this field is not applicable.

For an outline font, this field contains the minimum height that the font designer intended for this font. Note that this is not a restriction on the size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

FOCAMETRICS Field - usMaximumPointSize

usMaximumPointSize (SHORT)
Maximum point size.

For a bit-map font, this field is not applicable.

For an outline font, this field contains the maximum height that the font designer intended for this font. Note that this is not a restriction on the size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

FOCAMETRICS Field - usTypeFlags

usTypeFlags (SHORT)
Type indicators.

These flags contain the following information:

FM_TYPE_FIXED	Characters in the font have the same fixed width.
FM_TYPE_LICENSED	Licensed (protected) font.
FM_TYPE_KERNING	Font contains kerning information.
FM_TYPE_64K	Font is larger than 64KB (KB equals 1024 bytes) in size. Only one of the following bits (FM_TYPE_DBCS or FM_TYPE_MCBS) may be set. If both of those bits are false (not set), the font is for single-byte code pages.
FM_TYPE_DBCS	Font is for double-byte code pages.
FM_TYPE_MBCS	Font is for mixed single- and double-byte code pages.
FM_TYPE_FACETRUNC	Font *szFacename[32]* has been truncated.
FM_TYPE_FAMTRUNC	Font *szFamilyname[32]* has been truncated.

FOCAMETRICS Field - fsDefn

fsDefn (SHORT)
Definition indicators.

These indicators contain the following font definition data:

FM_DEFN_OUTLINE	Font is a vector (outline) font; otherwise, it is a bit map font.
FM_DEFN_GENERIC	Font is in a format that can be used by the GPI; otherwise, it is a device font.

FOCAMETRICS Field - fsSelectionFlags

fsSelectionFlags (SHORT)
Selection indicators.

These indicators contain information about the font patterns in the physical font.

Note: The flags do not reflect simulations applied to the physical font.

FM_SEL_ITALIC	TRUE indicates that this font is designed as an italic font.
FM_SEL_UNDERSCORE	TRUE indicates that this font is designed with underscores included in each character.
FM_SEL_NEGATIVE	TRUE indicates that this font is designed with the background and foreground reversed.
FM_SEL_OUTLINE	TRUE indicates that this font is designed with outline (hollow) characters.
FM_SEL_STRIKEOUT	TRUE indicates that this font is designed with an overstrike through each character.
FM_SEL_BOLD	TRUE indicates that this font is designed with bold characters.

FOCAMETRICS Field - fsCapabilities

fsCapabilities (SHORT)

Capabilities.

This attribute applies only to device fonts.

FM_CAP_NOMIX	Characters may not be mixed with graphics.
QUALITY	The most significant byte may contain the following numeric value:
	0 Undefined
	1 DP quality
	2 DP draft
	3 Near Letter Quality
	4 Letter Quality

FOCAMETRICS Field - ySubscriptXSize

ySubscriptXSize (SHORT)
Subscript X-size.

The recommended horizontal point size for subscripts for this font, in world coordinate units.

FOCAMETRICS Field - ySubscriptYSize

ySubscriptYSize (SHORT)
Subscript Y-size.

The recommended vertical point size for subscripts for this font, in world coordinate units.

FOCAMETRICS Field - ySubscriptXOffset

ySubscriptXOffset (SHORT)
Subscript X-offset.

The recommended baseline X-offset for subscripts for this font, in world coordinate units.

FOCAMETRICS Field - ySubscriptYOffset

ySubscriptYOffset (SHORT)
Subscript Y-offset.

The recommended baseline Y-offset for subscripts for this font, in world coordinate units.

FOCAMETRICS Field - ySuperscriptXSize

ySuperscriptXSize (SHORT)
Superscript X-size.

The recommended horizontal point size for superscripts for this font, in world coordinate units.

FOCAMETRICS Field - ySuperscriptYSize

ySuperscriptYSize (SHORT)
Superscript Y-size.

The recommended vertical point size for superscripts for this font, in world coordinate units.

FOCAMETRICS Field - ySuperscriptXOffset

ySuperscriptXOffset (SHORT)
Superscript X-offset.

The recommended baseline X-offset for superscripts for this font, in world coordinate units.

FOCAMETRICS Field - ySuperscriptYOffset

ySuperscriptYOffset (SHORT)
Superscript Y-offset.

The recommended baseline Y-offset for superscripts for this font, in world coordinate units.

FOCAMETRICS Field - yUnderscoreSize

yUnderscoreSize (SHORT)
Underscore size.

The width (thickness) of the underscore stroke, in world coordinate units. This describes the actual underscore in the font if FM_SEL_UNDERSCORE is also set. Otherwise, it describes what the graphics engine will simulate if underscore is requested in GpiCreateLogFont.

FOCAMETRICS Field - yUnderscorePosition

yUnderscorePosition (SHORT)
Underscore position.

The position of the underscore stroke from the baseline, in world coordinate units. This value describes the actual underscore in the font if FM_SEL_UNDERSCORE is also set. Otherwise, it describes what the graphics engine will simulate if underscore is requested in GpiCreateLogFont.

Note: Positive values mean below the baseline.

FOCAMETRICS Field - yStrikeoutSize

yStrikeoutSize (SHORT)
Strikeout size.

The width of the strikeout stroke, in world coordinate units. This value describes the actual strikeout in the font if FM_SEL_STRIKEOUT is also set. Otherwise, it describes what the graphics engine will simulate if overstrike is requested in GpiCreateLogFont.

FOCAMETRICS Field - yStrikeoutPosition

yStrikeoutPosition (SHORT)
Strikeout position.

The position of the strikeout stroke relative to the baseline, in world coordinate units. This value describes the actual strikeout in the font if FM_SEL_STRIKEOUT is also set. Otherwise, it describes what the graphics engine will simulate if overstrike is requested in GpiCreateLogFont.

FOCAMETRICS Field - usKerningPairs

usKerningPairs (SHORT)
Kerning pairs.

The number of kerning pairs in the kerning pair table.

FOCAMETRICS Field - sFamilyClass

sFamilyClass (SHORT)
Font family design classification.

This value contains a font class and its subclass.

FOCAMETRICS Field - pszDeviceNameOffset

pszDeviceNameOffset (PSZ)
Address where device name is stored.

FONTMETRICS

Font-metrics structure.

This structure is returned to applications on the GpiQueryFonts and GpiQueryFontMetrics calls and conveys information from the font creator to the application.

```
typedef struct _FONTMETRICS {
  CHAR       szFamilyname[FACESIZE];  /*  Family name. */
  CHAR       szFacename[FACESIZE];    /*  Face name. */
  USHORT     idRegistry;              /*  Registry identifier. */
  USHORT     usCodePage;              /*  Code page. */
  LONG       lEmHeight;               /*  Em height. */
  LONG       lXHeight;                /*  X height. */
  LONG       lMaxAscender;            /*  Maximum ascender. */
  LONG       lMaxDescender;           /*  Maximum descender. */
  LONG       lLowerCaseAscent;        /*  Lowercase ascent. */
  LONG       lLowerCaseDescent;       /*  Lowercase descent. */
  LONG       lInternalLeading;        /*  Internal leading. */
  LONG       lExternalLeading;        /*  External leading. */
  LONG       lAveCharWidth;           /*  Average character width. */
  LONG       lMaxCharInc;             /*  Maximum character increment. */
  LONG       lEmInc;                  /*  Em increment. */
  LONG       lMaxBaselineExt;         /*  Maximum baseline extent. */
  SHORT      sCharSlope;              /*  Character slope. */
  SHORT      sInlineDir;              /*  Inline direction. */
  SHORT      sCharRot;                /*  Character rotation. */
  USHORT     usWeightClass;           /*  Weight class. */
  USHORT     usWidthClass;            /*  Width class. */
  SHORT      sXDeviceRes;             /*  X-device resolution. */
  SHORT      sYDeviceRes;             /*  Y-device resolution. */
  SHORT      sFirstChar;              /*  First character. */
  SHORT      sLastChar;               /*  Last character. */
  SHORT      sDefaultChar;            /*  Default character. */
  SHORT      sBreakChar;              /*  Break character. */
  SHORT      sNominalPointSize;       /*  Nominal point size. */
  SHORT      sMinimumPointSize;       /*  Minimum point size. */
  SHORT      sMaximumPointSize;       /*  Maximum point size. */
  USHORT     fsType;                  /*  Type indicators. */
  USHORT     fsDefn;                  /*  Definition indicators. */
  USHORT     fsSelection;             /*  Selection indicators. */
  USHORT     fsCapabilities;          /*  Font capabilities. */
  LONG       lSubscriptXSize;         /*  Subscript x-size. */
  LONG       lSubscriptYSize;         /*  Subscript y-size. */
  LONG       lSubscriptXOffset;       /*  Subscript x-offset. */
  LONG       lSubscriptYOffset;       /*  Subscript y-offset. */
  LONG       lSuperscriptXSize;       /*  Superscript x-size. */
  LONG       lSuperscriptYSize;       /*  Superscript y-size. */
  LONG       lSuperscriptXOffset;     /*  Superscript x-offset. */
  LONG       lSuperscriptYOffset;     /*  Superscript y-offset. */
  LONG       lUnderscoreSize;         /*  Underscore size. */
  LONG       lUnderscorePosition;     /*  Underscore position. */
  LONG       lStrikeoutSize;          /*  Strikeout size. */
```

```
  LONG       lStrikeoutPosition;      /*  Strikeout position. */
  SHORT      sKerningPairs;           /*  Kerning pairs. */
  SHORT      sFamilyClass;            /*  Font family design classification. */
  LONG       lMatch;                  /*  Matched font identity. */
  LONG       FamilyNameAtom;          /*  Font family name atom. */
  LONG       FaceNameAtom;            /*  Font facename atom. */
  PANOSE     panose;                  /*  Panose font descriptor. */
} FONTMETRICS;

typedef FONTMETRICS *PFONTMETRICS;
```

FONTMETRICS Field - szFamilyname[FACESIZE]

szFamilyname[FACESIZE] (CHAR)
Family name.

The family name of the font that describes the basic appearance of the font, for example, Times New Roman** This string is null terminated, and therefore is limited to 31 characters in length. Longer names may be retrieved by using the *FamilyNameAtom* field to retrieve the full name from the System Atom table.

FONTMETRICS Field - szFacename[FACESIZE]

szFacename[FACESIZE] (CHAR)
Face name.

The typeface name that defines the particular font, for example, Times New Roman Bold Italic. This string is null terminated, and therefore is limited to 31 characters in length. Longer names may be retrieved by using the *FaceNameAtom* field to retrieve the full name from the System Atom table.

FONTMETRICS Field - idRegistry

idRegistry (USHORT)
Registry identifier.

The IBM registered number (or zero).

FONTMETRICS Field - usCodePage

usCodePage (USHORT)
Code page.

Defines the registered code page supported by the font. For example, the original IBM PC code page is 437. A value of 0 implies that the font may be used with any of the OS/2 supported code pages.

Where a font contains special symbols for which there is no registered code page, then code page 65400 is used.

FONTMETRICS Field - lEmHeight

lEmHeight (LONG)
Em height.

The height of the Em square in world coordinate units. This corresponds to the point size for the font.

FONTMETRICS Field - lXHeight

lXHeight (LONG)
X height.

The nominal height above the baseline for lowercase characters (ignoring ascenders) in world coordinate units.

FONTMETRICS Field - lMaxAscender

lMaxAscender (LONG)
Maximum ascender.

The maximum height above the baseline reached by any part of any symbol in the font in world coordinate units. This field may exceed *lEmHeight* .

FONTMETRICS Field - lMaxDescender

lMaxDescender (LONG)
Maximum descender.

The maximum depth below the baseline reached by any part of any symbol in the font in world coordinate units. This field may exceed *lEmHeight* .

FONTMETRICS Field - lLowerCaseAscent

lLowerCaseAscent (LONG)

Lowercase ascent.

The maximum height above the baseline reached by any part of any lowercase (Latin unaccented "a" through "z") symbol in the font in world coordinate units.

FONTMETRICS Field - lLowerCaseDescent

lLowerCaseDescent (LONG)
Lowercase descent.

The maximum depth below the baseline reached by any part of any lowercase (Latin unaccented "a" through "z") symbol in the font in world coordinate units.

FONTMETRICS Field - lInternalLeading

lInternalLeading (LONG)
Internal leading.

The amount of space which, when subtracted from *lMaxAscender* , gives a font-design dependent, but glyph-set independent, measure of the distance above the baseline that characters extend. This calculation approximates the visual top to a row of characters without actually looking at the characters in the row.

For optimum results, this field should be used by applications to position the first line of a block of text by subtracting it from *lMaxAscender* and positioning the baseline that distance below whatever is above the text.

Note: This does not guarantee that characters will not overwrite information above them, but does give a font designer's view of where to place the text. Collision should be tested for, and additional space allocated if necessary.

FONTMETRICS Field - lExternalLeading

lExternalLeading (LONG)
External leading.

The amount of guaranteed white space advised by the font designer to appear between adjacent rows of text. This value may be zero.

Note: The fonts built in to Presentation Manager have zero in this field.

FONTMETRICS Field - lAveCharWidth

lAveCharWidth (LONG)
Average character width.

This is determined by multiplying the width of each lowercase character by a constant, adding the products, and then dividing by

1000. The letters involved in this, plus their constants, are as follows:

Letter	Constant
a	64
b	14
c	27
d	35
e	100
f	20
g	14
h	42
i	63
j	3
k	6
l	35
m	20
n	56
o	56
p	17
q	4
r	49
s	56
t	71
u	31
v	10
w	18
x	3
y	18
z	2
space	166

Note: For fixed pitch fonts, this value will be the same as the (A width + B width + C width) escapement of each character.

FONTMETRICS Field - lMaxCharInc

lMaxCharInc (LONG)
Maximum character increment.

The maximum character increment for the font in world coordinate units.

FONTMETRICS Field - lEmInc

lEmInc (LONG)
Em increment.

The width of the Em square in world coordinate units. This corresponds to the point size of the font. When the horizontal device resolution equals the vertical device resolution this is equal to the em height.

FONTMETRICS Field - lMaxBaselineExt

lMaxBaselineExt (LONG)

Maximum baseline extent.

The maximum vertical space occupied by the font, in world coordinate units. This is the sum of *lMaxAscender* and *lMaxDescender* if both are positive. It is also the sum of *lInternalLeading* and *lEmHeight*.

One possible type of line spacing can be computed by adding *lMaxBaselineExt* to *lExternalLeading*. Such a line spacing, however, would be dependent on the glyph set included in the font. If a new version of the font should be made available, with new glyphs, then it is possible that this value will change because one of the new glyphs has gone above the previous *lMaxAscender* or below the previous *lMaxDescender*. More sophisticated applications will base line spacing on the point size (*lEmHeight*) of the font, which is an invariant of the font, multiplied by some factor (for example, 120%) plus any external leading.

This field may exceed *lEmHeight*.

FONTMETRICS Field - sCharSlope

sCharSlope (SHORT)
Character slope.

Defines the nominal slope for the characters of a font. The slope is defined in degrees increasing clockwise from the vertical. An *italic* font is an example of a font with a nonzero slope.

Note: The units for this metric are degrees and minutes, encoded as shown in the following example:

```
180 degrees 59 minutes would be represented as :

    < byte 1  >     < byte 2   >

    < Minutes >     < Degrees >

 0 1 1 1 0 1 1   0 1 0 1 1 0 1 0 0

     59 min         180 degrees
```

FONTMETRICS Field - sInlineDir

sInlineDir (SHORT)
Inline direction.

The direction in which the characters in the font are designed for viewing. The direction is defined in degrees increasing clockwise from the horizontal (left-to-right). Characters are added to a line of text in the inline direction.

Note: The units for this metric are degrees and minutes, encoded as shown in *sCharSlope*.

FONTMETRICS Field - sCharRot

sCharRot (SHORT)
Character rotation.

The rotation of the character glyphs with respect to the baseline, the angle increasing counter clockwise. This is the angle assigned by the font designer.

Note: The units for this metric are degrees and minutes, encoded as shown in *sCharSlope* .

FONTMETRICS Field - usWeightClass

usWeightClass (USHORT)
Weight class.

Indicates the visual weight (thickness of strokes) of the characters in the font:

Value	Description
1	Ultra-light
2	Extra-light
3	Light
4	Semi-light
5	Medium (normal)
6	Semi-bold
7	Bold
8	Extra-bold
9	Ultra-bold

FONTMETRICS Field - usWidthClass

usWidthClass (USHORT)
Width class.

Indicates the relative aspect ratio of the characters of the font in relation to the normal aspect ratio for this type of font:

Value	Description	% of normal width
1	Ultra-condensed	50
2	Extra-condensed	62.5
3	Condensed	75
4	Semi-condensed	87.5
5	Medium (normal)	100
6	Semi-expanded	112.5
7	Expanded	125
8	Extra-expanded	150
9	Ultra-expanded	200

FONTMETRICS Field - sXDeviceRes

sXDeviceRes (SHORT)
X-device resolution.

For bit-map fonts this is the resolution in the X direction of the intended target device, measured in pels per inch.

For outline fonts this is the number of notional units in the X direction of the Em square, measured in notional units per Em. (Notional units are the units in which the outline is defined.)

FONTMETRICS Field - sYDeviceRes

sYDeviceRes (SHORT)
Y-device resolution.

For bit-map fonts this is the resolution in the Y direction of the intended target device, measured in pels per inch.

For outline fonts this is the number of notional units in the Y direction of the Em square, measured in notional units per Em. (Notional units are the units in which the outline is defined.)

FONTMETRICS Field - sFirstChar

sFirstChar (SHORT)
First character.

The code point of the first character in the font.

FONTMETRICS Field - sLastChar

sLastChar (SHORT)
Last character.

The code point of the last character in the font, expressed as an offset from *sFirstChar* .

All code points between the first and last character specified must be supported by the font.

FONTMETRICS Field - sDefaultChar

sDefaultChar (SHORT)
Default character.

The code point that is used if a code point outside the range supported by the font is used, expressed as an offset from *sFirstChar* .

FONTMETRICS Field - sBreakChar

sBreakChar (SHORT)
Break character.

The code point that represents the "space" or "break" character for this font, expressed as an offset from *sFirstChar*. For example, if the first character is the space in code page 850, *sFirstChar* = 32, and *sBreakChar* = 0.

FONTMETRICS Field - sNominalPointSize

sNominalPointSize (SHORT)
Nominal point size.

For a bit-map font, this field contains the height of the font.

For an outline font, this field contains the height intended by the font designer. For example, some fonts are designed for text use in which case a value of 120 (12 point) would probably be placed in this field, whereas other fonts are designed for "display" use ("display" is typographer's terminology for larger sizes). This is not the only size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

FONTMETRICS Field - sMinimumPointSize

sMinimumPointSize (SHORT)
Minimum point size.

For a bit-map font, this field does not apply. For an outline font, this field contains the minimum height intended by the font designer. Note that this is not a restriction of the size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

FONTMETRICS Field - sMaximumPointSize

sMaximumPointSize (SHORT)
Maximum point size.

For a bit-map font, this field does not apply.

For an outline font, this field contains the maximum height intended by the font designer. Note that this is not a restriction of the size at which the font can be used.

Measured in decipoints (a decipoint is 1/720th of an inch).

FONTMETRICS Field - fsType

fsType (USHORT)
Type indicators.

This field contains the following information:

FM_TYPE_FIXED	Characters in the font have the same fixed width.
FM_TYPE_LICENSED	Licensed (protected) font.
FM_TYPE_KERNING	Font contains kerning information.
FM_TYPE_64K	Font is larger than 64KB (KB equals 1024 bytes) in size. If the following two bits are false, the font is for single-byte code pages. One of the bits may be set.
FM_TYPE_DBCS	Font is for double-byte code pages.
FM_TYPE_MBCS	Font is for mixed single- or double-byte code pages.

FM_TYPE_FACETRUNC
Font *szFacename* has been truncated.
FM_TYPE_FAMTRUNC
Font *szFamilyname* has been truncated.
FM_TYPE_ATOMS
The System Atom table atom values in *FamilyNameAtom* and in *FaceNameAtom* are valid.

FONTMETRICS Field - fsDefn

fsDefn (USHORT)
Definition indicators.

Contains the following font definition data:

FM_DEFN_OUTLINE	Font is a vector (outline) font; otherwise, it is a bit-map font.
FM_DEFN_GENERIC	Font is in a format that can be used by the GPI; otherwise, it is a device font.

FONTMETRICS Field - fsSelection

fsSelection (USHORT)
Selection indicators.

Contains information about the font patterns in the physical font.

Note: The flags do not reflect simulations applied to the physical font.

Possible values are:

FM_SEL_ITALIC	True indicates that this font is designed as an italic font.

FM_SEL_UNDERSCORE	TRUE indicates that this font is designed with underscores included in each character.
FM_SEL_NEGATIVE	TRUE indicates that this font is designed with the background and foreground reversed.
FM_SEL_OUTLINE	TRUE indicates that this font is designed with outline (hollow) characters.
FM_SEL_STRIKEOUT	TRUE indicates that this font is designed with an overstrike through each character.
FM_SEL_BOLD	TRUE indicates that this font is designed with bold characters.
FM_SEL_ISO9241_TESTED	This flag indicates that the font has been tested for compliance to ISO 9241. The presence of this flag doesn't indicate whether the font passed or failed, only that it was tested. **Note:** While the fonts were primarily tested for meeting the ISO standard, they have also been designed to meet the German standard DIN 66 234. Where the two standards differ, the fonts have been designed to meet the more stringent requirement.

FONTMETRICS Field - fsCapabilities

fsCapabilities (USHORT)
Font capabilities.

This attribute applies only to device fonts.

FM_CAP_NOMIX	Characters may not be mixed with graphics.
QUALITY	The most significant byte may contain the following numeric value: 0 Undefined 1 DP quality 2 DP draft 3 Near Letter Quality 4 Letter Quality

FONTMETRICS Field - lSubscriptXSize

lSubscriptXSize (LONG)
Subscript x-size.

The horizontal size recommended by the font designer for subscripts for this font in world coordinate units.

FONTMETRICS Field - lSubscriptYSize

lSubscriptYSize (LONG)
Subscript y-size.

The vertical size recommended by the font designer for subscripts for this font in world coordinate units.

FONTMETRICS Field - lSubscriptXOffset

lSubscriptXOffset (LONG)
Subscript x-offset.

The baseline x-offset recommended by the font designer for subscripts for this font in world coordinate units.

FONTMETRICS Field - lSubscriptYOffset

lSubscriptYOffset (LONG)
Subscript y-offset.

The baseline y-offset recommended by the font designer for subscripts for this font in world coordinate units.

Note: Positive numbers indicate an offset below the baseline.

FONTMETRICS Field - lSuperscriptXSize

lSuperscriptXSize (LONG)
Superscript x-size.

The horizontal size recommended by the font designer for superscripts for this font in world coordinate units.

FONTMETRICS Field - lSuperscriptYSize

lSuperscriptYSize (LONG)
Superscript y-size.

The vertical point size recommended by the font designer for superscripts for this font in world coordinate units.

FONTMETRICS Field - lSuperscriptXOffset

lSuperscriptXOffset (LONG)
Superscript x-offset.

The baseline x-offset recommended by the font designer for superscripts for this font in world coordinate units.

FONTMETRICS Field - lSuperscriptYOffset

lSuperscriptYOffset (LONG)
Superscript y-offset.

The baseline y-offset recommended by the font designer for superscripts for this font in world coordinate units.

FONTMETRICS Field - lUnderscoreSize

lUnderscoreSize (LONG)
Underscore size.

The width (thickness) of the underscore stroke in world coordinate units. This describes the actual underscore in the font if FM_SEL_UNDERSCORE is also set. Otherwise it describes what the engine will simulate if underscore is requested in GpiCreateLogFont.

FONTMETRICS Field - lUnderscorePosition

lUnderscorePosition (LONG)
Underscore position.

The position of the underscore stroke from the baseline in world coordinate units. This describes the actual underscore in the font if FM_SEL_UNDERSCORE is also set. Otherwise it describes what the engine will simulate if underscore is requested in GpiCreateLogFont.

Note: Positive values indicate an offset below the baseline.

FONTMETRICS Field - lStrikeoutSize

lStrikeoutSize (LONG)
Strikeout size.

The width of the strikeout stroke in world coordinate units. This describes the actual underscore in the font if FM_SEL_STRIKEOUT is also set. Otherwise it describes what the engine will simulate if overstrike is requested in GpiCreateLogFont.

FONTMETRICS Field - lStrikeoutPosition

lStrikeoutPosition (LONG)
Strikeout position.

The position of the strikeout stroke relative to the baseline in world coordinate units. This describes the actual underscore in the font if FM_SEL_STRIKEOUT is also set. Otherwise it describes what the engine will simulate if overstrike is requested in GpiCreateLogFont.

FONTMETRICS Field - sKerningPairs

sKerningPairs (SHORT)
Kerning pairs.

The number of kerning pairs in the kerning pair table.

FONTMETRICS Field - sFamilyClass

sFamilyClass (SHORT)
Font family design classification.

This value contains a font class and its subclass.

FONTMETRICS Field - lMatch

lMatch (LONG)
Matched font identity.

This uniquely identifies the font for a given device and device driver combination. A positive match number signifies that the font is a generic (engine) font while a negative number indicates a device font (a native or downloadable font). This value should not be used to identify a font across system boundaries.

FONTMETRICS Field - FamilyNameAtom

FamilyNameAtom (LONG)
Font family name atom.

This value contains the atom identifier for the font family name in the System Atom Table.

FONTMETRICS Field - FaceNameAtom

FaceNameAtom (LONG)
Font facename atom.

This value contains the atom identifier for the font face name in the System Atom Table.

FONTMETRICS Field - panose

panose (PANOSE)
Panose font descriptor.

This is the Panose descriptor identifying the visual characteristics of the font.

GRADIENTL

Direction-vector structure.

```
typedef struct _GRADIENTL {
  LONG     x;  /*  X-component of direction. */
  LONG     y;  /*  Y-component of direction. */
} GRADIENTL;

typedef GRADIENTL *PGRADIENTL;
```

GRADIENTL Field - x

x (LONG)
X-component of direction.

GRADIENTL Field - y

y (LONG)
Y-component of direction.

HAB

Anchor-block handle.

```
typedef LHANDLE HAB;
```

HBITMAP

Bit-map handle.

```
typedef LHANDLE HBITMAP;
```

HCINFO

Hardcopy-capabilities structure.

```
typedef struct _HCINFO {
  CHAR     szFormname[32];  /*  Form name. */
  LONG     cx;              /*  Width (left-to-right) in millimeters. */
  LONG     cy;              /*  Height (top-to-bottom) in millimeters. */
  LONG     xLeftClip;       /*  Left-clip limit in millimeters. */
  LONG     yBottomClip;     /*  Bottom-clip limit in millimeters. */
  LONG     xRightClip;      /*  Right-clip limit in millimeters. */
  LONG     yTopClip;        /*  Top-clip limit in millimeters. */
  LONG     xPels;           /*  Number of pels between left-clip and right-clip limits. */
  LONG     yPels;           /*  Number of pels between bottom-clip and top-clip limits. */
  LONG     flAttributes;    /*  Attributes of the form identifier. */
} HCINFO;

typedef HCINFO *PHCINFO;
```

HCINFO Field - szFormname[32]

szFormname[32] (CHAR)
Form name.

HCINFO Field - cx

cx (LONG)
Width (left-to-right) in millimeters.

HCINFO Field - cy

cy (LONG)
Height (top-to-bottom) in millimeters.

HCINFO Field - xLeftClip

xLeftClip (LONG)
Left-clip limit in millimeters.

HCINFO Field - yBottomClip

yBottomClip (LONG)
Bottom-clip limit in millimeters.

HCINFO Field - xRightClip

xRightClip (LONG)
Right-clip limit in millimeters.

HCINFO Field - yTopClip

yTopClip (LONG)
Top-clip limit in millimeters.

HCINFO Field - xPels

xPels (LONG)
Number of pels between left-clip and right-clip limits.

HCINFO Field - yPels

yPels (LONG)
Number of pels between bottom-clip and top-clip limits.

HCINFO Field - flAttributes

flAttributes (LONG)
Attributes of the form identifier.

HCAPS_SELECTABLE
The form is installed on the printer as given by the printer properties dialog. It is available from an alternate form source without operator intervention. If the form does not have this bit set and is used (if the user selects it), a "forms mismatch" error is generated by the printer object.

HCAPS_CURRENT
The form is the one currently selected by the DevOpenDC DEVOPENSTRUC *pdriv* field (the job properties).

HDC

Device-context handle.

```
typedef LHANDLE HDC;
```

HMF

Metafile handle.

```
typedef LHANDLE HMF;
```

HMODULE

Module handle.

```
typedef LHANDLE HMODULE;
```

HPAL

Palette handle.

```
typedef LHANDLE HPAL;
```

HPS

Presentation-space handle.

```
typedef LHANDLE HPS;
```

HRGN

Region handle.

```
typedef LHANDLE HRGN;
```

IMAGEBUNDLE

Image-attributes bundle structure.

```
typedef struct _IMAGEBUNDLE {
  LONG       lColor;         /*  Image foreground color. */
  LONG       lBackColor;     /*  Image background color. */
  USHORT     usMixMode;      /*  Image foreground-mix mode. */
  USHORT     usBackMixMode;  /*  Image background-mix mode. */
} IMAGEBUNDLE;

typedef IMAGEBUNDLE *PIMAGEBUNDLE;
```

IMAGEBUNDLE Field - lColor

lColor (LONG)
Image foreground color.

IMAGEBUNDLE Field - lBackColor

lBackColor (LONG)
Image background color.

IMAGEBUNDLE Field - usMixMode

usMixMode (USHORT)
Image foreground-mix mode.

IMAGEBUNDLE Field - usBackMixMode

usBackMixMode (USHORT)
Image background-mix mode.

KERNINGPAIRS

[Need description.]

```
typedef struct _KERNINGPAIRS {
  SHORT     sFirstChar;      /*  [Need definition.] */
  SHORT     sSecondChar;     /*  [Need definition.] */
  LONG      lKerningAmount;  /*  [Need definition.] */
} KERNINGPAIRS;

typedef KERNINGPAIRS *PKERNINGPAIRS;
```

KERNINGPAIRS Field - sFirstChar

sFirstChar (SHORT)
[Need definition.]

KERNINGPAIRS Field - sSecondChar

sSecondChar (SHORT)
[Need definition.]

KERNINGPAIRS Field - lKerningAmount

lKerningAmount (LONG)
[Need definition.]

LINEBUNDLE

Line-attributes bundle structure.

```
typedef struct _LINEBUNDLE {
  LONG       lColor;         /*  Line foreground color. */
  LONG       lBackColor;     /*  Line background color. */
  USHORT     usMixMode;      /*  Line foreground-mix mode. */
  USHORT     usBackMixMode;  /*  Line background-mix mode. */
  FIXED      fxWidth;        /*  Line width. */
  LONG       lGeomWidth;     /*  Geometric line width. */
  USHORT     usType;         /*  Line type. */
  USHORT     usEnd;          /*  Line end. */
  USHORT     usJoin;         /*  Line join. */
  USHORT     usReserved;     /*  Reserved. */
} LINEBUNDLE;

typedef LINEBUNDLE *PLINEBUNDLE;
```

LINEBUNDLE Field - lColor

lColor (LONG)
Line foreground color.

LINEBUNDLE Field - lBackColor

lBackColor (LONG)
Line background color.

LINEBUNDLE Field - usMixMode

usMixMode (USHORT)
Line foreground-mix mode.

LINEBUNDLE Field - usBackMixMode

usBackMixMode (USHORT)
Line background-mix mode.

LINEBUNDLE Field - fxWidth

fxWidth (FIXED)
Line width.

LINEBUNDLE Field - lGeomWidth

lGeomWidth (LONG)
Geometric line width.

LINEBUNDLE Field - usType

usType (USHORT)
Line type.

LINEBUNDLE Field - usEnd

usEnd (USHORT)
Line end.

LINEBUNDLE Field - usJoin

usJoin (USHORT)

Line join.

LINEBUNDLE Field - usReserved

usReserved (USHORT)
Reserved.

LONG

Signed integer in the range -2 147 483 648 through 2 147 483 647.

```
#define LONG long
```

Note: Where this data type represents a graphic coordinate in world or model space, its value is restricted to -134 217 728 through 134 217 727.

A graphic coordinate in device or screen coordinates is restricted to -32 768 through 32 767.

The value of a graphic coordinate may be further restricted by any transforms currently in force, including the positioning of the origin of the window on the screen. In particular, coordinates in world or model space must not generate coordinate values after transformation (that is, in device or screen space) outside the range -32 768 through 32 767.

MARKERBUNDLE

Marker-attributes bundle structure.

```
typedef struct _MARKERBUNDLE {
  LONG       lColor;         /*  Marker foreground color. */
  LONG       lBackColor;     /*  Marker background color. */
  USHORT     usMixMode;      /*  Marker foreground-mix mode. */
  USHORT     usBackMixMode;  /*  Marker background-mix mode. */
  USHORT     usSet;          /*  Marker set. */
  USHORT     usSymbol;       /*  Marker symbol. */
  SIZEF      sizfxCell;      /*  Marker cell. */
} MARKERBUNDLE;

typedef MARKERBUNDLE *PMARKERBUNDLE;
```

MARKERBUNDLE Field - lColor

lColor (LONG)
Marker foreground color.

MARKERBUNDLE Field - lBackColor

lBackColor (LONG)
Marker background color.

MARKERBUNDLE Field - usMixMode

usMixMode (USHORT)
Marker foreground-mix mode.

MARKERBUNDLE Field - usBackMixMode

usBackMixMode (USHORT)
Marker background-mix mode.

MARKERBUNDLE Field - usSet

usSet (USHORT)
Marker set.

MARKERBUNDLE Field - usSymbol

usSymbol (USHORT)
Marker symbol.

MARKERBUNDLE Field - sizfxCell

sizfxCell (SIZEF)

Marker cell.

MATRIXLF

Matrix-elements structure.

```
typedef struct _MATRIXLF {
  FIXED     fxM11;  /*  First element of first row. */
  FIXED     fxM12;  /*  Second element of first row. */
  LONG      lM13;   /*  Third element of first row. */
  FIXED     fxM21;  /*  First element of second row. */
  FIXED     fxM22;  /*  Second element of second row. */
  LONG      lM23;   /*  Third element of second row. */
  LONG      lM31;   /*  First element of third row. */
  LONG      lM32;   /*  Second element of third row. */
  LONG      lM33;   /*  Third element of third row. */
} MATRIXLF;

typedef MATRIXLF *PMATRIXLF;
```

MATRIXLF Field - fxM11

fxM11 (FIXED)
First element of first row.

MATRIXLF Field - fxM12

fxM12 (FIXED)
Second element of first row.

MATRIXLF Field - lM13

lM13 (LONG)
Third element of first row.

MATRIXLF Field - fxM21

fxM21 (FIXED)
First element of second row.

MATRIXLF Field - fxM22

fxM22 (FIXED)
Second element of second row.

MATRIXLF Field - lM23

lM23 (LONG)
Third element of second row.

MATRIXLF Field - lM31

lM31 (LONG)
First element of third row.

MATRIXLF Field - lM32

lM32 (LONG)
Second element of third row.

MATRIXLF Field - lM33

lM33 (LONG)
Third element of third row.

MPARAM

A 4-byte message-dependent parameter structure.

```
typedef VOID *MPARAM;
```

Certain elements of information, placed into the parameters of a message, have data types that do not use all four bytes of this data type. The rules governing these cases are:

BOOL	The value is contained in the low word and the high word is 0.
SHORT	The value is contained in the low word and its sign is extended into the high word.
USHORT	The value is contained in the low word and the high word is 0.
NULL	The entire four bytes are 0.

The structure of this data type depends on the message. For details, see the description of the particular message.

MRESULT

A 4-byte message-dependent reply parameter structure.

```
typedef VOID *MRESULT;
```

Certain elements of information, placed into the parameters of a message, have data types that do not use all four bytes of this data type. The rules governing these cases are:

BOOL	The value is contained in the low word and the high word is 0.
SHORT	The value is contained in the low word and its sign is extended into the high word.
USHORT	The value is contained in the low word and the high word is 0.
NULL	The entire four bytes are 0.

The structure of this data type depends on the message. For details, see the description of the particular message.

PANOSE

The *Panose* field in the font metrics will allow for quantitative descriptions of the visual properties of font faces. The PANOSE definition contains ten digits, each of which currently describes up to sixteen variations.

```
typedef struct _PANOSE {
  BYTE     bFamilyType;       /*  Family kind. */
  BYTE     bSerifStyle;       /*  Serif style. */
  BYTE     bWeight;           /*  Weight. */
  BYTE     bProportion;       /*  Proportion. */
  BYTE     bContrast;         /*  Contrast. */
  BYTE     bStrokeVariation;  /*  Stroke Variation. */
  BYTE     bArmStyle;         /*  Arm Style. */
  BYTE     bLetterform;       /*  Letterform. */
  BYTE     bMidline;          /*  Midline. */
  BYTE     bXHeight;          /*  X-Height. */
  BYTE     fbPassedISO;       /*  Font passed ISO test. */
  BYTE     fbFailedISO;       /*  Font failed ISO test. */
} PANOSE;

typedef PANOSE *PPANOSE;
```

PANOSE Field - bFamilyType

bFamilyType (BYTE)
Family kind.

0	Any
1	No Fit
2	Text and Display
3	Script
4	Decorative
5	Pictorial

PANOSE Field - bSerifStyle

bSerifStyle (BYTE)
Serif style.

0	Any
1	No Fit
2	Cove
3	Obtuse Cove
4	Square Cove
5	Obtuse Square Cove
6	Square
7	Thin
8	Bone
9	Exaggerated
10	Triangle
11	Normal Sans
12	Obtuse Sans
13	Perp Sans
14	Flared
15	Rounded

PANOSE Field - bWeight

bWeight (BYTE)
Weight.

0	Any
1	No Fit
2	Very Light
3	Light
4	Thin
5	Book
6	Medium
7	Demi
8	Bold
9	Heavy
10	Black
11	Nord

PANOSE Field - bProportion

bProportion (BYTE)
Proportion.

0	Any
1	No Fit
2	Old Style
3	Modern
4	Even Width
5	Expanded
6	Condensed
7	Very Expanded
8	Very Condensed
9	Monospaced

PANOSE Field - bContrast

bContrast (BYTE)
Contrast.

0	Any
1	No Fit
2	None
3	Very Low
4	Low
5	Medium Low
6	Medium
7	Medium High
8	High
9	Very High

PANOSE Field - bStrokeVariation

bStrokeVariation (BYTE)
Stroke Variation.

0	Any
1	No Fit
2	Gradual/Diagonal
3	Gradual/Transitional
4	Gradual/Vertical
5	Gradual/Horizontal
6	Rapid/Vertical
7	Rapid/Horizontal
8	Instant/Vertical

PANOSE Field - bArmStyle

bArmStyle (BYTE)
Arm Style.

0	Any
1	No Fit
2	Straight Arms/Horizontal
3	Straight Arms/Wedge
4	Straight Arms/Vertical
5	Straight Arms/Single Serif
6	Straight Arms/Double Serif
7	Non-Straight Arms/Horizontal
8	Non-Straight Arms/Wedge
9	Non-Straight Arms/Vertical
10	Non-Straight Arms/Single Serif
11	Non-Straight Arms/Double Serif

PANOSE Field - bLetterform

bLetterform (BYTE)
Letterform.

0	Any
1	No Fit
2	Normal/Contact
3	ONormal/Weighted
4	ONormal/Boxed
5	ONormal/Flattened
6	ONormal/Rounded
7	ONormal/Off Center
8	ONormal/Square
9	Oblique/Contact
10	Oblique/Weighted
11	Oblique/Boxed
12	Oblique/Flattened
13	Oblique/Rounded
14	Oblique/Off Center
15	Oblique/Square

PANOSE Field - bMidline

bMidline (BYTE)
Midline.

0	Any
1	No Fit
2	Standard/Trimmed
3	Standard/Pointed
4	Standard/Serifed
5	High/Trimmed
6	High/Pointed
7	High/Serifed
8	Constant/Trimmed
9	Constant/Pointed

10	Constant/Serifed
11	Low/Trimmed
12	Low/Pointed
13	Low/Serifed

PANOSE Field - bXHeight

bXHeight (BYTE)
X-Height.

0	Any
1	No Fit
2	Constant/Small
3	Constant/Standard
4	Constant/Large
5	Ducking/Small
6	Ducking/Standard
7	Ducking/Large

PANOSE Field - fbPassedISO

fbPassedISO (BYTE)
Font passed ISO test.

The following flags indicate those displays and resolutions at which the font complied with ISO 9241:

FM_ISO_9518_640
FM_ISO_9515_640
FM_ISO_9515_1024
FM_ISO_9517_640
FM_ISO_9517_1024

PANOSE Field - fbFailedISO

fbFailedISO (BYTE)
Font failed ISO test.

The following flags indicate those displays and resolutions at which the font did not comply with ISO 9241:

FM_ISO_9518_640
FM_ISO_9515_640
FM_ISO_9515_1024
FM_ISO_9517_640
FM_ISO_9517_1024

PBUNDLE

Points to a bundle data area.

```
typedef PVOID PBUNDLE;
```

PCH

Pointer to a character string.

```
typedef unsigned char *PCH;
```

POLYGON

Polygon structure.

```
typedef struct _POLYGON {
  ULONG       ulPoints;  /*  Number of points in array. */
  PPOINTL     aPointl;   /*  Array of points. */
} POLYGON;

typedef POLYGON *PPOLYGON;
```

POLYGON Field - ulPoints

ulPoints (ULONG)
Number of points in array.

POLYGON Field - aPointl

aPointl (PPOINTL)
Array of points.

POINTL

Point structure (long integers).

```
typedef struct _POINTL {
  LONG     x;  /*  X-coordinate. */
  LONG     y;  /*  Y-coordinate. */
} POINTL;

typedef POINTL *PPOINTL;
```

POINTL Field - x

x (LONG)
X-coordinate.

POINTL Field - y

y (LONG)
Y-coordinate.

PCSZ

Pointer to a constant null-terminated string.

```
typedef const char *PCSZ;
```

PSZ

Pointer to a null-terminated string.

If you are using C++ **, you may need to use PCSZ.

```
typedef unsigned char *PSZ;
```

PRQINFO3

Print-queue information structure.

This structure is used at information levels 3 and 4.

```
typedef struct _PRQINFO3 {
  PSZ           pszName;        /*  Queue name. */
  USHORT        uPriority;      /*  Queue priority. */
  USHORT        uStartTime;     /*  Minutes after midnight when queue becomes active. */
  USHORT        uUntilTime;     /*  Minutes after midnight. when queue ceases to be active. */
  USHORT        fsType;         /*  Queue type. */
  PSZ           pszSepFile;     /*  Separator-page file. */
  PSZ           pszPrProc;      /*  Default queue-processor. */
  PSZ           pszParms;       /*  Queue parameters. */
  PSZ           pszComment;     /*  Queue description. */
  USHORT        fsStatus;       /*  Queue status. */
  USHORT        cJobs;          /*  Number of jobs in queue. */
  PSZ           pszPrinters;    /*  Print devices connected to queue. */
  PSZ           pszDriverName;  /*  Default device driver. */
  PDRIVDATA     pDriverData;    /*  Default queue job properties. */
} PRQINFO3;

typedef PRQINFO3 *PPRQINFO3;
```

PRQINFO3 Field - pszName

pszName (PSZ)
Queue name.

The maximum length of the name in the network case is 256 (including one byte for zero termination).

PRQINFO3 Field - uPriority

uPriority (USHORT)
Queue priority.

The range is 1 through 9, with 1 being the highest queue priority.

The default job priority (DefJobPrio) is determined from:
DefJobPrio=100-(10* *uPriority*).

If a job is added with **PRJ_NO_PRIORITY** specified, DefJobPrio is used. If a default priority higher than the default job priority is specified, the default job priority is used. If a default priority lower than the default is specified, the specified job priority is used.

PRQ_DEF_PRIORITY
: Default priority

PRQ_MAX_PRIORITY
: Highest priority

PRQ_MIN_PRIORITY
: Minimum priority

PRQ_NO_PRIORITY
: No priority.

PRQINFO3 Field - uStartTime

uStartTime (USHORT)

Minutes after midnight when queue becomes active.

For example, the value 75 represents 1:15 a.m.

If *uStartTime* and *uUntilTime* are both 0, the print queue is always available.

PRQINFO3 Field - uUntilTime

uUntilTime (USHORT)
Minutes after midnight. when queue ceases to be active.

For example, the value 1200 represents 8 p.m.

If *uUntilTime* and *uStartTime* are both 0, the print queue is always available.

PRQINFO3 Field - fsType

fsType (USHORT)
Queue type.

PRQ3_TYPE_RAW
Data is always enqueued in the device specific format.
PRQ3_TYPE_BYPASS
Allows the spooler to bypass the queue processor and send data directly to the Printer Driver. Setting this bit allows the spooler to print jobs of type PM_Q_RAW while they are still being spooled.
PRQ3_TYPE_APPDEFAULT
This bit is set for the application default queue only.

PRQINFO3 Field - pszSepFile

pszSepFile (PSZ)
Separator-page file.

The path and file name of a separator-page file on the target computer.

This file contains formatting information for the page or pages to be used between print jobs. A relative path name is taken as relative to the current spool directory. A NULL string indicates no separator page.

PRQINFO3 Field - pszPrProc

pszPrProc (PSZ)
Default queue-processor.

PRQINFO3 Field - pszParms

pszParms (PSZ)
Queue parameters.

This can be any text string or a NULL string.

PRQINFO3 Field - pszComment

pszComment (PSZ)
Queue description.

A NULL string results in no comment. The maximum length is 48 characters (including one byte for the null terminator).

PRQINFO3 Field - fsStatus

fsStatus (USHORT)
Queue status.

PRQ3_PAUSED
Queue is paused (held).
PRQ3_PENDING
Queue is pending deletion.

PRQINFO3 Field - cJobs

cJobs (USHORT)
Number of jobs in queue.

PRQINFO3 Field - pszPrinters

pszPrinters (PSZ)
Print devices connected to queue.

This cannot be NULL.

PRQINFO3 Field - pszDriverName

pszDriverName (PSZ)
Default device driver.

PRQINFO3 Field - pDriverData

pDriverData (PDRIVDATA)
Default queue job properties.

Note: An application can use *pszDriverName* , *pDriverData* , *pszPrProc* , and *pszParms* to construct a valid DevOpenDC call based only on the queue name.

PVOID

Pointer to a data type of undefined format.

```
typedef VOID *PVOID;
```

RECTL

Rectangle structure.

```
typedef struct _RECTL {
  LONG     xLeft;    /*  X-coordinate of left-hand edge of rectangle. */
  LONG     yBottom;  /*  Y-coordinate of bottom edge of rectangle. */
  LONG     xRight;   /*  X-coordinate of right-hand edge of rectangle. */
  LONG     yTop;     /*  Y-coordinate of top edge of rectangle. */
} RECTL;

typedef RECTL *PRECTL;
```

RECTL Field - xLeft

xLeft (LONG)
X-coordinate of left-hand edge of rectangle.

RECTL Field - yBottom

yBottom (LONG)
Y-coordinate of bottom edge of rectangle.

RECTL Field - xRight

xRight (LONG)
X-coordinate of right-hand edge of rectangle.

RECTL Field - yTop

yTop (LONG)
Y-coordinate of top edge of rectangle.

RGNRECT

Region-rectangle structure.

```
typedef struct _RGNRECT {
  ULONG     ircStart;     /*  Rectangle number from which to start enumerating. */
  ULONG     crc;          /*  Number of rectangles that can be returned. */
  ULONG     crcReturned;  /*  Number of rectangles returned. */
  ULONG     ulDirection;  /*  Direction in which the returned rectangles are to be ordered. */
} RGNRECT;

typedef RGNRECT *PRGNRECT;
```

RGNRECT Field - ircStart

ircStart (ULONG)
Rectangle number from which to start enumerating.

Numbering starts from 1.

RGNRECT Field - crc

crc (ULONG)
Number of rectangles that can be returned.

This must be 1 or greater.

RGNRECT Field - crcReturned

crcReturned (ULONG)
Number of rectangles returned.

A value of less than *crc* indicates that there are no more rectangles to enumerate.

RGNRECT Field - ulDirection

ulDirection (ULONG)
Direction in which the returned rectangles are to be ordered.

This ordering uses the leading edge of a rectangle:

RECTDIR_LFRT_TOPBOT	Left-to-right, top-to-bottom
RECTDIR_RTLF_TOPBOT	Right-to-left, top-to-bottom
RECTDIR_LFRT_BOTTOP	Left-to-right, bottom-to-top
RECTDIR_RTLF_BOTTOP	Right-to-left, bottom-to-top

RGB

RGB color value.

```
typedef struct _RGB {
  BYTE     bBlue;   /*  Blue component of the color definition. */
  BYTE     bGreen;  /*  Green component of the color definition. */
  BYTE     bRed;    /*  Red component of the color definition. */
} RGB;

typedef RGB *PRGB;
```

RGB Field - bBlue

bBlue (BYTE)
Blue component of the color definition.

RGB Field - bGreen

bGreen (BYTE)
Green component of the color definition.

RGB Field - bRed

bRed (BYTE)
Red component of the color definition.

RGB2

RGB color value.

```
typedef struct _RGB2 {
  BYTE     bBlue;      /*  Blue component of the color definition. */
  BYTE     bGreen;     /*  Green component of the color definition. */
  BYTE     bRed;       /*  Red component of the color definition. */
  BYTE     fcOptions;  /*  Entry options. */
} RGB2;

typedef RGB2 *PRGB2;
```

RGB2 Field - bBlue

bBlue (BYTE)
Blue component of the color definition.

RGB2 Field - bGreen

bGreen (BYTE)

Green component of the color definition.

RGB2 Field - bRed

bRed (BYTE)
Red component of the color definition.

RGB2 Field - fcOptions

fcOptions (BYTE)
Entry options.

These can be ORed together if required:

PC_RESERVED	The color entry is reserved for animating color with the palette manager.
PC_EXPLICIT	The low-order word of the color table entry designates a physical palette slot. This allows an application to show the actual contents of the device palette as realized for other logical palettes. This does not prevent the color in the slot from being changed for any reason.

SIZEF

Size structure (FIXED values).

```
typedef struct _SIZEF {
  FIXED     cx;  /*  Width. */
  FIXED     cy;  /*  Height. */
} SIZEF;

typedef SIZEF *PSIZEF;
```

SIZEF Field - cx

cx (FIXED)
Width.

SIZEF Field - cy

cy (FIXED)
Height.

SIZEL

Size structure (LONG values).

```
typedef struct _SIZEL {
  LONG     cx;  /*  Width. */
  LONG     cy;  /*  Height. */
} SIZEL;

typedef SIZEL *PSIZEL;
```

SIZEL Field - cx

cx (LONG)
Width.

SIZEL Field - cy

cy (LONG)
Height.

SHORT

Signed integer in the range -32 768 through 32 767.

```
#define SHORT short
```

STR8

String of 8 characters.

```
typedef CHAR STR8[8];
```

ULONG

Unsigned integer in the range 0 through 4 294 967 295.

```
typedef unsigned long ULONG;
```

USHORT

Unsigned integer in the range 0 through 65 535.

```
typedef unsigned short USHORT;
```

Graphics Orders

This section describes the format of the graphics orders.

Graphics orders are used in the following circumstances:

- Using GpiGetData or GpiPutData functions for bulk transfer of part or all of graphics segment data (unless this is simply being copied without being changed).
- Editing segments with GpiQueryElement and GpiElement.
- Generating metafiles (other than through the Presentation Manager API), or examining their contents. The data part of Graphics Data structured fields within the metafile (see "Metafile Data Format" in the *Presentation Manager Programming Reference*) consists of graphics orders.

When primitive or attribute functions (plus certain other functions) are specified at the programming interface, and the drawing mode (see GpiSetDrawingMode) is set to **drawandretain**, graphics orders are constructed and placed in the current graphics segment. One API call often causes a single order to be generated. Sometimes, however, several orders are necessary: an example of this is where a GpiPolyLine call is issued, which specifies more strokes than there is room for, in a single order.

In either case, the order or orders generated by a single API call comprise a single **element**, unless the application specifically starts an element using the GpiBeginElement function. In this case the element consists of all of the orders generated between this and the following GpiEndElement function. A GpiQueryElement function returns the orders that comprise an element; the application may edit these, and return them to the segment with GpiElement. The Begin Element - End Element orders that surround a multi-order element in the segment are never passed between the application and the system on GpiQueryElement and GpiElement functions.

No double word or word alignment can be assumed for orders either within segments or during editing.

Introduction to Graphics Orders

In the retain and draw-and-retain drawing modes, specific GPI functions (primitive-drawing and attribute-setting functions, plus some others) cause graphics orders to be stored in the current segment. A graphics order is a sequence of one or more bytes of data that describe a graphics function. There is typically a one-to-one correspondence between a GPI function and a graphics order. You do not need to understand the various formats and contents of the graphics orders, unless:

- You are using GpiGetData or GpiPutData for bulk transfer of data that you want to edit.
- You are simply copying data from one segment to another.
- You are using GpiElement to add data to a segment, or GpiQueryElement to retrieve data from a segment.
- You are examining the contents of a metafile.

Both the graphics orders and the metafile structure are described in the Presentation Manager. This appendix describes the header file PMORD.H, which has been provided to allow you to manipulate the graphics orders more easily.

The Graphics-Orders Header File (PMORD.H)

A set of helper constants, macros, and structures has been provided to help you decode and encode graphics orders. These items are defined in the header file PMORD.H.

There are four types of graphics orders. The first byte of each order, regardless of the graphics-order type, is the order code itself, which either partially or completely describes what follows. Depending on the order type, the graphics order can contain further information.

The four types of graphics order are:

1-Byte Order

The 1-byte order comprises a single byte:

```
BYTE 1 : order code.
```

2-Byte Order

The 2-byte order consists of two bytes:

```
BYTE 1 : order code
BYTE 2 : associated value.
```

Long Order

The long order can comprise up to 257 bytes of information:

```
BYTE 1     : order code
BYTE 2     : length of order (0 to 255)
BYTE 3-257 : associated value bytes
             depending on the order code.
```

There is a special long order (Escape) where:

```
BYTE 3     : escape type
BYTE 4     : escape identifier
BYTE 5-257 : associated value bytes
             depending on the escape
             identifier.
```

Very Long Order

The very long order can comprise up to 65537 bytes of information:

```
BYTE 1       : order code
BYTE 2       : order qualifier
BYTE 3       : length of order
               (most significant byte)
BYTE 4       : length of order
               (least significant byte -
                length of order is 0 to 65535)
BYTE 5-65537 : associated value bytes
               depending on the order
               qualifier.
```

There is a special very long order (Escape) where :

```
BYTE 5       : escape type
BYTE 6       : escape identifier
BYTE 7-65537 : associated value bytes
               depending on the escape
               identifier.
```

Decoding Graphics Orders

The recommended way of decoding a buffer of graphics orders (in C language) is to use a pointer to address the first byte of the buffer, and then retrieve the graphics order it contains. To discover which of the four types of order you have, use the following macros:

- BYTE_ORDER (1-byte order)
- SHORT_ORDER (2-byte order)
- LONG_ORDER (long order)
- VLONG_ORDER (very long order).

These macros are defined in the header file PMORD.H. Each macro processes a single byte of data and returns a Boolean value (zero or nonzero). A zero value means that the order is not of that type. When you know the graphics-order type, you can establish the length of the order, and add the length to the pointer. You can then retrieve the next order in the buffer, and repeat the process until all data has been retrieved.

You can decode the graphics-order data itself by providing a routine for each of the order types, or a routine for each individual order:

- For a 1-byte graphic order, the decoding routine should simply return a length of 1.
- For a 2-byte graphic order, the decoding routine can use the overlay structure ORDER to decode the data. The routine should return a length of 2.
- For a long order, the decoding routine can use the overlay structure LORDER to decode the data. The length of the data is a variable value.
- For a very long order, the decoding routine can use the overlay structure VORDER to decode the data. The length of the data is a variable value.

The overlay structures ORDER, LORDER, and VORDER are defined in the header file PMORD.H.

You can build graphics orders using the same structures and order types that are used for decoding graphics orders.

Naming Conventions

The names of the graphics-order codes are in the form OCODE_Gxxx. The Gxxx abbreviation is the name of the individual order, and can be used for types, structures, and constants directly related to that order. In the header file, there is a comment on the same line as each of the orders that describes the order. For example, the Begin Area order (GBAR) is described in the header file as follows:

```
#define OCODE_GBAR     0x68  /* Begin area    */
#define GBAR_BOUNDARY  0xC0
```

Note:

In some structures, an S or an L is added to the name to differentiate between the short-coordinate form (16-bit) and the long-coordinate form (32-bit). For example, the Set Arc Parameters order (GSAP) is as follows:

```
#define OCODE_GSAP     0x22
#define OCODE_GPSAP    0x62
```

```
typedef struct _ORDERS_GSAP {
  SHORT  p;
  SHORT  q;
  SHORT  r;
  SHORT  s;
} ORDERS_GSAP;

typedef struct _ORDERL_GSAP {
  LONG   p;
  LONG   q;
  LONG   r;
  LONG   s;
} ORDERL_GSAP;
```

In this example, the structures ORDERS_GSAP and ORDERL_GSAP are shared by GSAP (set arc parameters) and GPSAP (push and set arc parameters). As a rule, there is structure sharing between the set and push-and-set forms of graphics orders.

There is structure-sharing between the current-position and the given-position forms of some orders. For example, the orders GCARC (arc at current position) and GARC (arc at given position) share a structure.

Arc at a Given Position/Arc at Current Position

Arc at a Given Position/Arc at Current Position - Syntax

This order constructs an arc starting at a given position.

Arc at a Given Position (GARC)
X'C6' (LEN, P0, P1, P2)

Arc at Current Position (GCARC)
X'86' (LEN, P1, P2)

GARC/GCARC Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GARC/GCARC Parameter - P0

P0 (GPOINT)
Coordinate data of start point.

This parameter is only present in a Arc at a Given Position order.

GARC/GCARC Parameter - P1

P1 (GPOINT)
Coordinate data of intermediate point.

GARC/GCARC Parameter - P2

P2 (GPOINT)
Coordinate data of end point.

Arc at a Given Position/Arc at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of start point.

This parameter is only present in a Arc at a Given Position order.

P1 (GPOINT)
Coordinate data of intermediate point.

P2 (GPOINT)
Coordinate data of end point.

Arc at a Given Position/Arc at Current Position - Topics

Select an item:
Syntax
Parameters
Glossary

Begin Area

Begin Area - Syntax

This order indicates the start of a set of primitives that define an area boundary.

Begin Area (GBAR)
X'68' (FLAGS)

GBAR Parameter - FLAGS

FLAGS
Internal flags.

RES1 (GBIT1)
Reserved for migration:

1	Only valid value.

BOUNDARY (GBIT1)
Boundary-line draw indicator:

0	Do not draw boundary lines
1	Draw boundary lines.

INSIDE (GBIT1)
Mode shading:

0	Alternate mode
1	Winding mode.

RES2 (GBIT5)
Reserved value, must be 0.

Begin Area - Parameters

FLAGS
Internal flags.

RES1 (GBIT1)
Reserved for migration:

1	Only valid value.

BOUNDARY (GBIT1)
Boundary-line draw indicator:

0	Do not draw boundary lines
1	Draw boundary lines.

INSIDE (GBIT1)
Mode shading:

0	Alternate mode
1	Winding mode.

RES2 (GBIT5)
Reserved value, must be 0.

Begin Area - Topics

Select an item:
Syntax
Parameters
Glossary

Begin Element

Begin Element - Syntax

This order indicates the beginning of a set of primitives that define an element.

Begin Element (GBEL)
X'D2' (LEN, TYPE, DESCR)

GBEL Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GBEL Parameter - TYPE

TYPE (GLONG)
Element type code.

Possible values are described in the following list:

0x0000FD01	Line bundle
0x0000FD02	Character bundle
0x0000FD03	Marker bundle
0x0000FD04	Area bundle
0x0000FD05	Image bundle
0x00000007	Call segment
0x00000081	Polyline
0x00000085	Polyfillet
0x000000A4	Polyfillet sharp
0x000000A5	Polyspline

0x00000082	Polymarker
0x00000087	Full arc
0x00000091	Image
0x000000B1	Character string at current position
0x000000F1	Character string at given position
0x81xxxxxx-0xFFxxxxxx	Indicates user defined elements
Other	Reserved values.

GBEL Parameter - DESCR

DESCR (GUNDF)
Element description data.

This is optional.

Begin Element - Parameters

LEN (GLENGTH1)
Length of following data.

TYPE (GLONG)
Element type code.

Possible values are described in the following list:

0x0000FD01	Line bundle
0x0000FD02	Character bundle
0x0000FD03	Marker bundle
0x0000FD04	Area bundle
0x0000FD05	Image bundle
0x00000007	Call segment
0x00000081	Polyline
0x00000085	Polyfillet
0x000000A4	Polyfillet sharp
0x000000A5	Polyspline
0x00000082	Polymarker
0x00000087	Full arc
0x00000091	Image
0x000000B1	Character string at current position
0x000000F1	Character string at given position
0x81xxxxxx-0xFFxxxxxx	Indicates user defined elements
Other	Reserved values.

DESCR (GUNDF)
Element description data.

This is optional.

Begin Element - Topics

Select an item:
Syntax

Parameters
Glossary

Begin Image at Given Position/Begin Image at Current Position

Begin Image at Given Position/Begin Image at Current Position - Syntax

These orders identify the start of an image definition at a given position or at the current position.

Begin Image at Given Position (GBIMG)
X'D1' (LEN, P0, FORMAT, RES, WIDTH, HEIGHT)

Begin Image at Current Position (GCBIMG)
X'91' (LEN, FORMAT, RES, WIDTH, HEIGHT)

GBIMG/GCBIMG Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x06	Only valid value.

GBIMG/GCBIMG Parameter - P0

P0 (GPOINT)
Point at which the image is to be placed.

This parameter is only present in a Begin Image at Given Position order.

GBIMG/GCBIMG Parameter - FORMAT

FORMAT (GBIT8)
Format of the image data.

0x00 One bit in the data represents one image point on the usable area.

GBIMG/GCBIMG Parameter - RES

RES (GBIT8)
Reserved.

0x00 Only valid value.

GBIMG/GCBIMG Parameter - WIDTH

WIDTH (GUSHORT370)
Width of the image data.

This is the width in pels.

0x00-0x07 Valid range of values.

GBIMG/GCBIMG Parameter - HEIGHT

HEIGHT (GUSHORT370)
Height of the image data.

This is the height in pels.

Begin Image at Given Position/Begin Image at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

0x06 Only valid value.

P0 (GPOINT)
Point at which the image is to be placed.

This parameter is only present in a Begin Image at Given Position order.

FORMAT (GBIT8)
Format of the image data.

0x00 One bit in the data represents one image point on the usable area.

RES (GBIT8)
Reserved.

0x00 Only valid value.

WIDTH (GUSHORT370)
Width of the image data.

This is the width in pels.

0x00-0x07 Valid range of values.

HEIGHT (GUSHORT370)
Height of the image data.

This is the height in pels.

Begin Image at Given Position/Begin Image at Current Position - Topics

Select an item:
Syntax
Parameters
Glossary

Begin Path

Begin Path - Syntax

This order sets the drawing process into path state.

Begin Path (GBPTH)
X'D0' (LEN, RES, PTHID)

GBPTH Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x06 Only valid value.

GBPTH Parameter - RES

RES (GBIT16)
Reserved.

0x0000	Only valid value.

GBPTH Parameter - PTHID

PTHID (GLONG)
Path identifier.

0x00000001-0xFFFFFFFF	Valid path identifiers.

Begin Path - Parameters

LEN (GLENGTH1)
Length of following data.

0x06	Only valid value.

RES (GBIT16)
Reserved.

0x0000	Only valid value.

PTHID (GLONG)
Path identifier.

0x00000001-0xFFFFFFFF	Valid path identifiers.

Begin Path - Topics

Select an item:
Syntax
Parameters
Glossary

Bezier Curve at Given Position/Bezier Curve at Current Poition

Bezier Curve at Given Position/Bezier Curve at Current Poition - Syntax

This order generates a curve that starts at a given position.

Bezier Curve at Given Position (GBEZ)
X'E5' (LEN, P0, P1, P2, P3, P4, P5, P6, PN-2, PN-1, PN)

Bezier Curve at Current Poition (GCBEZ)
X'A5' (LEN, P1, P2, P3, P4, P5, P6, PN-2, PN-1, PN)

GBEZ/GCBEZ Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GBEZ/GCBEZ Parameter - P0

P0 (GPOINT)
Coordinate data of first curve start.

This parameter is only present in a Bezier Curve at Given Position order.

GBEZ/GCBEZ Parameter - P1

P1 (GPOINT)
Coordinate data of first curve, first control point.

GBEZ/GCBEZ Parameter - P2

P2 (GPOINT)
Coordinate data of first curve, second control point.

GBEZ/GCBEZ Parameter - P3

P3 (GPOINT)
Coordinate data of first curve end.

GBEZ/GCBEZ Parameter - P4

P4 (GPOINT)
Coordinate data of second curve, first control point

GBEZ/GCBEZ Parameter - P5

P5 (GPOINT)
Coordinate data of second curve, second control point

GBEZ/GCBEZ Parameter - P6

P6 (GPOINT)
Coordinate data of second curve end.

GBEZ/GCBEZ Parameter - PN-2

PN-2 (GPOINT)
Coordinate data of final curve, first control point

GBEZ/GCBEZ Parameter - PN-1

PN-1 (GPOINT)
Coordinate data of final curve, second control point

GBEZ/GCBEZ Parameter - PN

PN (GPOINT)
Coordinate data of final curve end.

Bezier Curve at Given Position/Bezier Curve at Current Poition - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of first curve start.

This parameter is only present in a Bezier Curve at Given Position order.

P1 (GPOINT)
Coordinate data of first curve, first control point.

P2 (GPOINT)
Coordinate data of first curve, second control point.

P3 (GPOINT)
Coordinate data of first curve end.

P4 (GPOINT)
Coordinate data of second curve, first control point

P5 (GPOINT)
Coordinate data of second curve, second control point

P6 (GPOINT)
Coordinate data of second curve end.

PN-2 (GPOINT)
Coordinate data of final curve, first control point

PN-1 (GPOINT)
Coordinate data of final curve, second control point

PN (GPOINT)
Coordinate data of final curve end.

Bezier Curve at Given Position/Bezier Curve at Current Poition - Topics

Select an item:
Syntax
Parameters
Glossary

Bitblt

Bitblt - Syntax

This order copies a rectangle of a bit map into DOCS.

Bitblt (GBBLT)
X'D6' (LEN, FLAGS, MIX, BMID, TRANS, P1, P2, SOURCE1X, SOURCE1Y, SOURCE2X, SOURCE2Y)

GBBLT Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GBBLT Parameter - FLAGS

FLAGS (GBIT16)
Reserved.

0x0000	Only valid value.

GBBLT Parameter - MIX

MIX (GBIT16)
Mix mode.

Values are:

0x00CC	Source.
0x00C0	Source and pattern.
0x00CA	Source where pattern1
0x000C	Source where pattern0
0x00E2	Pattern where source1
0x00B8	Pattern where source0
Other	Reserved values.

GBBLT Parameter - BMID

BMID (GHBITMAP)
Bit-map identifier.

GBBLT Parameter - TRANS

TRANS (GBIT32)
Transfer mode.

Values are:

0x00000000	OR
0x01000000	AND
0x02000000	Ignore
Other	Reserved values.

GBBLT Parameter - P1

P1 (GPOINT)
Target rectangle bottom-left corner.

GBBLT Parameter - P2

P2 (GPOINT)
Target rectangle top-right corner.

GBBLT Parameter - SOURCE1X

SOURCE1X (GLONG)
Source rectangle bottom-left corner, x-coordinate.

GBBLT Parameter - SOURCE1Y

SOURCE1Y (GLONG)
Source rectangle bottom-left corner, y-coordinate.

GBBLT Parameter - SOURCE2X

SOURCE2X (GLONG)
Source rectangle top-right corner, x-coordinate.

GBBLT Parameter - SOURCE2Y

SOURCE2Y (GLONG)
Source rectangle top-right corner, y-coordinate.

Bitblt - Parameters

LEN (GLENGTH1)
Length of following data.

FLAGS (GBIT16)
Reserved.

0x0000	Only valid value.

MIX (GBIT16)
Mix mode.

Values are:

0x00CC	Source.
0x00C0	Source and pattern.
0x00CA	Source where pattern1
0x000C	Source where pattern0
0x00E2	Pattern where source1
0x00B8	Pattern where source0
Other	Reserved values.

BMID (GHBITMAP)
Bit-map identifier.

TRANS (GBIT32)
Transfer mode.

Values are:

0x00000000	OR
0x01000000	AND
0x02000000	Ignore
Other	Reserved values.

P1 (GPOINT)
Target rectangle bottom-left corner.

P2 (GPOINT)
Target rectangle top-right corner.

SOURCE1X (GLONG)
Source rectangle bottom-left corner, x-coordinate.

SOURCE1Y (GLONG)
Source rectangle bottom-left corner, y-coordinate.

SOURCE2X (GLONG)
Source rectangle top-right corner, x-coordinate.

SOURCE2Y (GLONG)
Source rectangle top-right corner, y-coordinate.

Bitblt - Topics

Select an item:
Syntax
Parameters
Glossary

Box at Given Position/Box at Current Position

Box at Given Position/Box at Current Position - Syntax

This order defines a box with square or round corners, drawn with its first corner at a given position.

Box at Given Position (GBOX)
X'C0' (LEN, CONTROL, RES, P0, P1, HAXIS, VAXIS)

Box at Current Position (GCBOX)
X'80' (LEN, CONTROL, RES, P1, HAXIS, VAXIS)

GBOX/GCBOX Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GBOX/GCBOX Parameter - CONTROL

CONTROL
Internal flags.

RES1 (GBIT1)
Reserved value, must be 0.

FILL (GBIT1)
Values:

0	No fill
1	Fill.

BOUNDARY (GBIT1)
Values:

0	No boundary
1	Boundary.

RES2 (GBIT5)
Reserved value, must be 0.

GBOX/GCBOX Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GBOX/GCBOX Parameter - P0

P0 (GPOINT)
Coordinate data of box origin.

This parameter is only present in a Box at Given Position order.

GBOX/GCBOX Parameter - P1

P1 (GPOINT)
Coordinate data of box corner.

GBOX/GCBOX Parameter - HAXIS

HAXIS (GROSOL)

Length of horizontal axis of ellipse.

GBOX/GCBOX Parameter - VAXIS

VAXIS (GROSOL)
Length of vertical axis of ellipse.

Box at Given Position/Box at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

CONTROL
Internal flags.

RES1 (GBIT1)
Reserved value, must be 0.

FILL (GBIT1)
Values:

0	No fill
1	Fill.

BOUNDARY (GBIT1)
Values:

0	No boundary
1	Boundary.

RES2 (GBIT5)
Reserved value, must be 0.

RES (GBIT8)
Reserved value, must be 0.

P0 (GPOINT)
Coordinate data of box origin.

This parameter is only present in a Box at Given Position order.

P1 (GPOINT)
Coordinate data of box corner.

HAXIS (GROSOL)
Length of horizontal axis of ellipse.

VAXIS (GROSOL)
Length of vertical axis of ellipse.

Box at Given Position/Box at Current Position - Topics

Select an item:

Syntax
Parameters
Glossary

Call Segment

Call Segment - Syntax

This order calls one segment from another.

Call Segment (GCALLS)
X'07' (LEN, RES, SEGNAME)

GCALLS Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x06 Only valid value.

GCALLS Parameter - RES

RES (GBIT16)
Reserved value, must be 0.

GCALLS Parameter - SEGNAME

SEGNAME (GLONG)
Name of segment that is to be called.

The name cannot be 0.

Call Segment - Parameters

LEN (GLENGTH1)
Length of following data.

0x06 Only valid value.

RES (GBIT16)
Reserved value, must be 0.

SEGNAME (GLONG)
Name of segment that is to be called.

The name cannot be 0.

Call Segment - Topics

Select an item:
Syntax
Parameters
Glossary

Character String at Given Position/Character String at Current Position

Character String at Given Position/Character String at Current Position - Syntax

These orders draw a character string at a given position or at the current position.

Character String at Given Position (GCHST)
X'C3' (LEN, P0, CP)

Character String at Current Position (GCCHST)
X'83' (LEN, CP)

GCHST/GCCHST Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GCHST/GCCHST Parameter - P0

P0 (GPOINT)
Point at which the character string is to be placed.

This parameter is only present in a Character String at Given Position order.

GCHST/GCCHST Parameter - CP

CP (GSTR)
Code points of each character in the string.

Character String at Given Position/Character String at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Point at which the character string is to be placed.

This parameter is only present in a Character String at Given Position order.

CP (GSTR)
Code points of each character in the string.

Character String at Given Position/Character String at Current Position - Topics

Select an item:
Syntax
Parameters
Glossary

Character String Extended at Given Position/Character String Extended at Current Position

Character String Extended at Given Position/Character String Extended at Current Position - Syntax

This order defines a character string to be drawn at a given position.

Character String Extended at Given Position (GCHSTE)
X'FEF0' (LEN1, P0, FLAGS, RES, P1, P2, LEN2, CP, PAD, VECT)

Character String Extended at Current Position (GCCHSTE)
X'FEB0' (LEN1, FLAGS, RES, P1, P2, LEN2, CP, PAD, VECT)

GCHSTE/GCCHSTE Parameter - LEN1

LEN1 (GLENGTH2)
Length of following data.

GCHSTE/GCCHSTE Parameter - P0

P0 (GPOINT)
Point at which the character string is to be placed.

This parameter is only present in a Character String Extended at Given Position order.

GCHSTE/GCCHSTE Parameter - FLAGS

FLAGS
Extra functions:

RECT (GBIT1)
Values:

0	Do not draw background rectangle
1	Draw background rectangle.

CLIP (GBIT1)
Values:

0	Do not clip to rectangle
1	Clip to rectangle.

RES1 (GBIT1)
Reserved value, must be 0.

LVCP (GBIT1)
Values:

0	Move current position
1	Leave current position.

RES2 (GBIT4)
Reserved value, must be 0.

GCHSTE/GCCHSTE Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GCHSTE/GCCHSTE Parameter - P1

P1 (GPOINT)
Coordinate data of rectangle corner.

GCHSTE/GCCHSTE Parameter - P2

P2 (GPOINT)
Coordinate data of rectangle corner.

GCHSTE/GCCHSTE Parameter - LEN2

LEN2 (GLENGTH2)
Length of code-point data.

GCHSTE/GCCHSTE Parameter - CP

CP (GSTR)
Code-point data.

GCHSTE/GCCHSTE Parameter - PAD

PAD (GBIT8)
Pad byte.

Only needs to be included if *CP* is an odd number of bytes.

GCHSTE/GCCHSTE Parameter - VECT

VECT (GROSOL)
IN (0) Vector of character increments.

VECT is a vector of **n** elements, where **n** is the number of code points present in the *CP* parameter.

Character String Extended at Given Position/Character String Extended at Current Position - Parameters

LEN1 (GLENGTH2)
Length of following data.

P0 (GPOINT)
Point at which the character string is to be placed.

This parameter is only present in a Character String Extended at Given Position order.

FLAGS
Extra functions:

RECT (GBIT1)
Values:

0	Do not draw background rectangle
1	Draw background rectangle.

CLIP (GBIT1)
Values:

0	Do not clip to rectangle
1	Clip to rectangle.

RES1 (GBIT1)
Reserved value, must be 0.

LVCP (GBIT1)
Values:

0	Move current position
1	Leave current position.

RES2 (GBIT4)
Reserved value, must be 0.

RES (GBIT8)
Reserved value, must be 0.

P1 (GPOINT)

Coordinate data of rectangle corner.

P2 (GPOINT)
Coordinate data of rectangle corner.

LEN2 (GLENGTH2)
Length of code-point data.

CP (GSTR)
Code-point data.

PAD (GBIT8)
Pad byte.

Only needs to be included if *CP* is an odd number of bytes.

VECT (GROSOL)
IN (0) Vector of character increments.

VECT is a vector of **n** elements, where **n** is the number of code points present in the *CP* parameter.

Character String Extended at Given Position/Character String Extended at Current Position - Topics

Select an item:
Syntax
Parameters
Glossary

Character String Move at Given Position/Character String Move at Current Position

Character String Move at Given Position/Character String Move at Current Position - Syntax

This order draws a character string starting from a given position and moves the current position to the end of the string.

Character String Move at Given Position (GCHSTM)
X'F1' (LEN, P0, CP)

Character String Move at Current Position (GCCHSTM)
X'B1' (LEN, CP)

GCHSTM/GCCHSTM Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GCHSTM/GCCHSTM Parameter - P0

P0 (GPOINT)
Point at which the character string is to be placed.

This parameter is only present in a Character String Move at Given Position order.

GCHSTM/GCCHSTM Parameter - CP

CP (GSTR)
Code points of each character in the string.

Character String Move at Given Position/Character String Move at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Point at which the character string is to be placed.

This parameter is only present in a Character String Move at Given Position order.

CP (GSTR)
Code points of each character in the string.

Character String Move at Given Position/Character String Move at Current Position - Topics

Select an item:
Syntax
Parameters
Glossary

Close Figure

Close Figure - Syntax

This order delimits the end of a closed figure.

Close Figure (GCLFIG)
X'7D' (RES)

GCLFIG Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

Close Figure - Parameters

RES (GBIT8)
Reserved value, must be 0.

Close Figure - Topics

Select an item:
Syntax
Parameters
Glossary

Comment

Comment - Syntax

This order enables data to be stored within a segment.

Comment (GCOMT)
X'01' (LEN, DATA[LEN])

GCOMT Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GCOMT Parameter - DATA[LEN]

DATA[LEN] (GBIT8)
Comment data.

Comment - Parameters

LEN (GLENGTH1)
Length of following data.

DATA[LEN] (GBIT8)
Comment data.

Comment - Remarks

This order is treated as a no-operation.

Comment - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

End Area

End Area - Syntax

This order indicates the end of a set of primitives that define an area boundary.

End Area (GEAR)
X'60' (LEN, DATA[LEN])

GEAR Parameter - LEN

LEN (GLENGTH1)
Length of following data. It is normally 0.

GEAR Parameter - DATA[LEN]

DATA[LEN] (GBIT8)
Reserved value, must be 0.

End Area - Parameters

LEN (GLENGTH1)
Length of following data. It is normally 0.

DATA[LEN] (GBIT8)
Reserved value, must be 0.

End Area - Topics

Select an item:
Syntax
Parameters
Glossary

End Element

End Element - Syntax

This order identifies the end of a set of primitives that define an element.

End Element (GEEL)
X'49' (RES)

GEEL Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

End Element - Parameters

RES (GBIT8)
Reserved value, must be 0.

End Element - Topics

Select an item:
Syntax
Parameters
Glossary

End Image

End Image - Syntax

This order identifies the end of an image definition.

End Image (GEIMG)
X'93' (LEN, DATA[LEN])

GEIMG Parameter - LEN

LEN (GLENGTH1)
Length of following data. It is normally 0.

GEIMG Parameter - DATA[LEN]

DATA[LEN] (GBIT8)
Reserved value, must be 0.

End Image - Parameters

LEN (GLENGTH1)
Length of following data. It is normally 0.

DATA[LEN] (GBIT8)
Reserved value, must be 0.

End Image - Topics

Select an item:
Syntax
Parameters
Glossary

End of Symbol Definition

End of Symbol Definition - Syntax

This order indicates the end of a set of orders defining a graphics symbol.

```
End of Symbol Definition (GESD)
X'FF'
```

End of Symbol Definition - Remarks

This order is only valid in the context of symbol definitions.

End of Symbol Definition - Topics

Select an item:
Syntax
Remarks
Glossary

End Path

End Path - Syntax

This order ends the definition of a path.

End Path (GEPTH)
X'7F' (RES)

GEPTH Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

End Path - Parameters

RES (GBIT8)
Reserved value, must be 0.

End Path - Topics

Select an item:
Syntax
Parameters
Glossary

End Prolog

End Prolog - Syntax

This order indicates the end of the prolog of a segment.

End Prolog (GEPROL)
X'3E' (RES)

GEPROL Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

End Prolog - Parameters

RES (GBIT8)
Reserved value, must be 0.

End Prolog - Topics

Select an item:
Syntax
Parameters
Glossary

Escape

Escape - Syntax

This order provides facilities for registered and unregistered escape functions.

Escape (GESCP)
X'D5' (LEN, TYPE, RID, PARMS)

GESCP Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GESCP Parameter - TYPE

TYPE (GBIT8)
Type identifier:

80	Registered value
Other	All other values are unregistered.

GESCP Parameter - RID

RID (GBIT8)
Registered identifier:

01	Set pel.
02	BITBLT function.
03	Flood fill function.
04	Draw bits function.

GESCP Parameter - PARMS

PARMS (GSTR)
Parameters of escape.

Escape - Parameters

LEN (GLENGTH1)
Length of following data.

TYPE (GBIT8)
Type identifier:

80	Registered value
Other	All other values are unregistered.

RID (GBIT8)
Registered identifier:

01	Set pel.
02	BITBLT function.
03	Flood fill function.
04	Draw bits function.

PARMS (GSTR)
Parameters of escape.

Escape - Topics

Select an item:
Syntax
Parameters
Glossary

Extended Escape

Extended Escape - Syntax

This order provides facilities for registered and unregistered escape functions.

Extended Escape (GEESCP)
X'FED5' (LEN, TYPE, RID, PARMS)

GEESCP Parameter - LEN

LEN (GLENGTH2)
Length of following data.

GEESCP Parameter - TYPE

TYPE (GBIT8)
Type identifier:

0x80	Registered value
Other	All other values are unregistered.

GEESCP Parameter - RID

RID (GUCHAR)
Registered identifier.

No registered extended escapes are used by the OS/2* operating system.

GEESCP Parameter - PARMS

PARMS (GSTR)
Parameters of escape.

Extended Escape - Parameters

LEN (GLENGTH2)
Length of following data.

TYPE (GBIT8)
Type identifier:

0x80	Registered value
Other	All other values are unregistered.

RID (GUCHAR)
Registered identifier.

No registered extended escapes are used by the OS/2* operating system.

PARMS (GSTR)
Parameters of escape.

Extended Escape - Topics

Select an item:
Syntax
Parameters
Glossary

Fill Path

Fill Path - Syntax

This order fills the interior of the specified path.

Fill Path (GFPTH)
X'D7' (LEN, FLAGS, RES, PTHID)

GFPTH Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x06 Only valid value.

GFPTH Parameter - FLAGS

FLAGS
Extra functions:

RES1 (GBIT1)
Reserved value, must be 0.

INSIDE (GBIT1)
Values:

0 Alternate mode
1 Winding mode.

MOD (GBIT1)
Values:

0 Do not modify before filling
1 Modify path before filling.

RES2 (GBIT5)
Reserved value, must be 0.

GFPTH Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GFPTH Parameter - PTHID

PTHID (GLONG)
Path identifier.

0x00000001-0xFFFFFFFF Valid path identifiers.

Fill Path - Parameters

LEN (GLENGTH1)
Length of following data.

0x06 Only valid value.

FLAGS
Extra functions:

RES1 (GBIT1)
Reserved value, must be 0.

INSIDE (GBIT1)
Values:

0 Alternate mode
1 Winding mode.

MOD (GBIT1)
Values:

0 Do not modify before filling
1 Modify path before filling.

RES2 (GBIT5)
Reserved value, must be 0.

RES (GBIT8)
Reserved value, must be 0.

PTHID (GLONG)
Path identifier.

0x00000001-0xFFFFFFFF Valid path identifiers.

Fill Path - Topics

Select an item:
Syntax
Parameters
Glossary

Fillet at Given Position/Fillet at Current Position

Fillet at Given Position/Fillet at Current Position - Syntax

These orders draw a curved line tangential to a specified set of straight lines, at the given position or at the current position.

Fillet at Given Position (GFLT)
X'C5' (LEN, P0, P1, P2, PN)

Fillet at Current Position (GCFLT)
X'85' (LEN, P1, P2, PN)

GFLT/GCFLT Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GFLT/GCFLT Parameter - P0

P0 (GPOINT)
Coordinate data of line start.

This parameter is only present in a Fillet at Given Position order.

GFLT/GCFLT Parameter - P1

P1 (GPOINT)
Coordinate data of first line end.

GFLT/GCFLT Parameter - P2

P2 (GPOINT)
Coordinate data of second line end.

GFLT/GCFLT Parameter - PN

PN (GPOINT)
Coordinate data of final line end.

Fillet at Given Position/Fillet at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of line start.

This parameter is only present in a Fillet at Given Position order.

P1 (GPOINT)
Coordinate data of first line end.

P2 (GPOINT)
Coordinate data of second line end.

PN (GPOINT)
Coordinate data of final line end.

Fillet at Given Position/Fillet at Current Position - Topics

Select an item:

Syntax
Parameters
Glossary

Full Arc at Given Position/Full Arc at Current Position

Full Arc at Given Position/Full Arc at Current Position - Syntax

This order constructs a full circle or an ellipse, with the center at a given position.

Full Arc at Given Position (GFARC)
X'C7' (LEN, P0, M)

Full Arc at Current Position (GCFARC)
X'87' (LEN, M)

GFARC/GCFARC Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GFARC/GCFARC Parameter - P0

P0 (GPOINT)
Coordinate data of the center of the circle/ellipse.

This parameter is only present in a Full Arc at Given Position order.

GFARC/GCFARC Parameter - M

M (GROFUFS)
Multiplier.

Full Arc at Given Position/Full Arc at Current Position -

Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of the center of the circle/ellipse.

This parameter is only present in a Full Arc at Given Position order.

M (GROFUFS)
Multiplier.

Full Arc at Given Position/Full Arc at Current Position - Topics

Select an item:
Syntax
Parameters
Glossary

Image Data

Image Data - Syntax

This order provides bit data for an image.

Image Data (GIMD)
X'92' (LEN, DATA[LEN])

GIMD Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GIMD Parameter - DATA[LEN]

DATA[LEN] (GBIT8)
Image data.

Image Data - Parameters

LEN (GLENGTH1)
Length of following data.

DATA[LEN] (GBIT8)
Image data.

Image Data - Topics

Select an item:
Syntax
Parameters
Glossary

Label

Label - Syntax

This order is used to label an element within a segment.

Label (GLBL)
X'D3' (LEN, LDATA)

GLBL Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x04	Only valid value.

GLBL Parameter - LDATA

LDATA (GLONG)
Label value.

Label - Parameters

LEN (GLENGTH1)
Length of following data.

0x04 Only valid value.

LDATA (GLONG)
Label value.

Label - Topics

Select an item:
Syntax
Parameters
Glossary

Line at Given Position/Line at Current Position

Line at Given Position/Line at Current Position - Syntax

This order defines one or more connected straight lines, drawn from the given position.

Line at Given Position (GLINE)
X'C1' (LEN, P0, P1, PN)

Line at Current Position (GCLINE)
X'81' (LEN, P1, PN)

GLINE/GCLINE Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GLINE/GCLINE Parameter - P0

P0 (GPOINT)
Coordinate data of line start.

This parameter is only present in a Line at Given Position order.

GLINE/GCLINE Parameter - P1

P1 (GPOINT)
Coordinate data of first line end.

GLINE/GCLINE Parameter - PN

PN (GPOINT)
Coordinate data of final line end.

Line at Given Position/Line at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of line start.

This parameter is only present in a Line at Given Position order.

P1 (GPOINT)
Coordinate data of first line end.

PN (GPOINT)
Coordinate data of final line end.

Line at Given Position/Line at Current Position - Topics

Select an item:

Syntax
Parameters
Glossary

Marker at Given Position/Marker at Current Position

Marker at Given Position/Marker at Current Position - Syntax

This order draws the current marker symbol at one or more positions starting from a given position.

Marker at Given Position (GMRK)
X'C2' (LEN, P0, P1, PN)

Marker at Current Position (GCMRK)
X'82' (LEN, P1, PN)

GMRK/GCMRK Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GMRK/GCMRK Parameter - P0

P0 (GPOINT)
Coordinate data of first marker.

GMRK/GCMRK Parameter - P1

P1 (GPOINT)
Coordinate data of second marker.

GMRK/GCMRK Parameter - PN

PN (GPOINT)
Coordinate data of final marker.

Marker at Given Position/Marker at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of first marker.

P1 (GPOINT)
Coordinate data of second marker.

PN (GPOINT)
Coordinate data of final marker.

Marker at Given Position/Marker at Current Position - Topics

Select an item:
Syntax
Parameters
Glossary

Modify Path

Modify Path - Syntax

This order modifies the path according to the value of the mode.

Modify Path (GMPTH)
X'D8' (LEN, MODE, RES, PTHID)

GMPTH Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x06	Only valid value.

GMPTH Parameter - MODE

MODE (GBIT8)
Mode of path modification:

0x06	Stroke the path
Other	All other values are reserved.

GMPTH Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GMPTH Parameter - PTHID

PTHID (GLONG)
Path identifier.

0x00000001-0xFFFFFFFF Valid path identifiers.

Modify Path - Parameters

LEN (GLENGTH1)
Length of following data.

0x06	Only valid value.

MODE (GBIT8)
Mode of path modification:

0x06	Stroke the path
Other	All other values are reserved.

RES (GBIT8)
Reserved value, must be 0.

PTHID (GLONG)
Path identifier.

0x00000001-0xFFFFFFFF Valid path identifiers.

Modify Path - Topics

Select an item:
Syntax
Parameters
Glossary

No-Operation

No-Operation - Syntax

This order is a no-operation.

```
No-Operation (GNOP1)
X'00'
```

No-Operation - Topics

Select an item:
Syntax
Glossary

Outline Path

Outline Path - Syntax

This order draws the outline of the specified path.

Outline Path (GOPTH)
X'D4' (LEN, FLAGS, RES, PTHID)

GOPTH Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GOPTH Parameter - FLAGS

FLAGS (GBIT8)
Function flags:

0x00	Only valid value.

GOPTH Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GOPTH Parameter - PTHID

PTHID (GLONG)
Path identifier.

1	Only valid value.

Outline Path - Parameters

LEN (GLENGTH1)
Length of following data.

FLAGS (GBIT8)
Function flags:

0x00	Only valid value.

RES (GBIT8)
Reserved value, must be 0.

PTHID (GLONG)
Path identifier.

1 Only valid value.

Outline Path - Topics

Select an item:
Syntax
Parameters
Glossary

Partial Arc at Given Position/Partial Arc at Current Position

Partial Arc at Given Position/Partial Arc at Current Position - Syntax

This order draws a line from a given position to the start of an arc, and then draws the arc.

Partial Arc at Given Position (GPARC)
X'E3' (LEN, P0, P1, M, START, SWEEP)

Partial Arc at Current Position (GCPARC)
X'A3' (LEN, P1, M, START, SWEEP)

GPARC/GCPARC Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GPARC/GCPARC Parameter - P0

P0 (GPOINT)
Coordinate data of start of line.

This parameter is only present in a Partial Arc at Given Position order.

GPARC/GCPARC Parameter - P1

P1 (GPOINT)
Coordinate data of center of arc.

GPARC/GCPARC Parameter - M

M (GROFUFS)
Multiplier.

GPARC/GCPARC Parameter - START

START (GROF)
Start angle.

GPARC/GCPARC Parameter - SWEEP

SWEEP (GROF)
Sweep angle.

Partial Arc at Given Position/Partial Arc at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of start of line.

This parameter is only present in a Partial Arc at Given Position order.

P1 (GPOINT)
Coordinate data of center of arc.

M (GROFUFS)
Multiplier.

START (GROF)
Start angle.

SWEEP (GROF)
Sweep angle.

Partial Arc at Given Position/Partial Arc at Current Position - Topics

Select an item:
Syntax
Parameters
Glossary

Polygons

Polygons - Syntax

This order defines a set of polygons, which are optionally filled.

Polygons (GPOLYS)
X'F3' (LEN, FLAGS, COUNT, POLYS)

GPOLYS Parameter - LEN

LEN (GLENGTH2)
Length of following data.

GPOLYS Parameter - FLAGS

FLAGS
Internal flags.

INSIDE (GBIT1)
Mode shading:

0 Alternate mode.

1 Winding mode.

MODEL (GBIT1)
Drawing model:

0 The fill is inclusive of bottom right.
1 The fill is exclusive of bottom right.

RES2 (GBIT6)
Reserved value, must be 0.

GPOLYS Parameter - COUNT

COUNT (GUSHORT)
Number of polygons.

GPOLYS Parameter - POLYS

POLYS (GPOLYS)
Array of polygons.

Polygons - Parameters

LEN (GLENGTH2)
Length of following data.

FLAGS
Internal flags.

INSIDE (GBIT1)
Mode shading:

0 Alternate mode.
1 Winding mode.

MODEL (GBIT1)
Drawing model:

0 The fill is inclusive of bottom right.
1 The fill is exclusive of bottom right.

RES2 (GBIT6)
Reserved value, must be 0.

COUNT (GUSHORT)
Number of polygons.

POLYS (GPOLYS)
Array of polygons.

Polygons - Remarks

This order draws a set of polygons. For the first polygon the current position is the first point. For all subsequent polygons all points which define the polygon are given explicitly. The polygons are automatically closed if necessary.

The current position is set to the last point specified.

Polygons - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Pop

Pop - Syntax

This order enables data to be popped from the Segment Call Stack.

Pop (GPOP)
X'3F' (RES)

GPOP Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

Pop - Parameters

RES (GBIT8)
Reserved value, must be 0.

Pop - Remarks

The data is placed into an attribute or Drawing Process Control.

Pop - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Relative Line at Given Position/Relative Line at Current Position

Relative Line at Given Position/Relative Line at Current Position - Syntax

These orders define one or more connected straight lines, at the given position or at the current position.

Relative Line at Given Position (GRLINE)
X'E1' (LEN, P0, OFF0, OFF1, OFFN)

Relative Line at Current Position (GCRLINE)
X'A1' (LEN, OFF0, OFF1, OFFN)

GRLINE/GCRLINE Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GRLINE/GCRLINE Parameter - P0

P0 (GPOINT)
Coordinate data of line start.

This parameter is only present in a Relative Line at Given Position order.

GRLINE/GCRLINE Parameter - OFF0

OFF0 (GDELPOINT)
Offset data for first point.

This offset is to the first line end, relative to its start point.

GRLINE/GCRLINE Parameter - OFF1

OFF1 (GDELPOINT)
Offset data for second point.

This offset is to the second line end, relative to the first line end.

GRLINE/GCRLINE Parameter - OFFN

OFFN (GDELPOINT)
Offset data for final point.

This offset is to the nth line end, relative to the n-1th line end.

Relative Line at Given Position/Relative Line at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of line start.

This parameter is only present in a Relative Line at Given Position order.

OFF0 (GDELPOINT)
Offset data for first point.

This offset is to the first line end, relative to its start point.

OFF1 (GDELPOINT)
Offset data for second point.

This offset is to the second line end, relative to the first line end.

OFFN (GDELPOINT)
Offset data for final point.

This offset is to the nth line end, relative to the n-1th line end.

Relative Line at Given Position/Relative Line at Current Position - Remarks

The end point of each line is given as an offset from the start of the line, rather than as absolute coordinates.

Relative Line at Given Position/Relative Line at Current Position - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Segment Characteristics

Segment Characteristics - Syntax

This order provides the facility to set architected or user-defined characteristics for a segment.

Segment Characteristics (GSGCH)
X'04' (LEN, CBIT8, PARMS)

GSGCH Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSGCH Parameter - CBIT8

CBIT8 (GUCHAR)
Identification code for characteristics:

0x00-0x7F	Reserved for architected characteristics.
0x80-0xFF	Reserved for user-defined characteristics.

GSGCH Parameter - PARMS

PARMS (GSTR)
Parameters of characteristics.

Segment Characteristics - Parameters

LEN (GLENGTH1)
Length of following data.

CBIT8 (GUCHAR)
Identification code for characteristics:

0x00-0x7F	Reserved for architected characteristics.
0x80-0xFF	Reserved for user-defined characteristics.

PARMS (GSTR)
Parameters of characteristics.

Segment Characteristics - Remarks

The order is only valid in a root-segment prolog.

Segment Characteristics - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Arc Parameters/Push and Set Arc Parameters

Set Arc Parameters/Push and Set Arc Parameters - Syntax

These orders set, or push and set, the values of the current arc parameters.

Set Arc Parameters (GSAP)
X'22' (LEN, P, Q, R, S)

Push and Set Arc Parameters (GPSAP)
X'62' (LEN, P, Q, R, S)

GSAP/GPSAP Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSAP/GPSAP Parameter - P

P (GROSOL)
P-parameter of arc transform.

GSAP/GPSAP Parameter - Q

Q (GROSOL)
Q-parameter of arc transform.

GSAP/GPSAP Parameter - R

R (GROSOL)
R-parameter of arc transform.

GSAP/GPSAP Parameter - S

S (GROSOL)
S-parameter of arc transform.

Set Arc Parameters/Push and Set Arc Parameters - Parameters

LEN (GLENGTH1)
Length of following data.

P (GROSOL)
P-parameter of arc transform.

Q (GROSOL)
Q-parameter of arc transform.

R (GROSOL)
R-parameter of arc transform.

S (GROSOL)
S-parameter of arc transform.

Set Arc Parameters/Push and Set Arc Parameters - Remarks

The values of the current arc parameters are pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the current arc parameters to the values specified in the order.

The value of these parameters determines the shape of subsequent orders drawn using Arc at a Given Position/Arc at Current Position or Full Arc at Given Position/Full Arc at Current Position or Partial Arc at Given Position/Partial Arc at Current Position.

Set Arc Parameters/Push and Set Arc Parameters - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Background Color/Push and Set Background Color

Set Background Color/Push and Set Background Color - Syntax

These orders set, or push and set, the value of the current background color attribute.

Set Background Color (GSBCOL)
X'25' (LEN, COLOR)

Push and Set Background Color (GPSBCOL)
X'65' (LEN, COLOR)

GSBCOL/GPSBCOL Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x02	Only valid value.

GSBCOL/GPSBCOL Parameter - COLOR

COLOR (GBIT16)
Color-table index: Except for the special values, the values 0x0000 through 0xnnnn are allowed color indexes; that is, as many values as are allowed by the size of the LCT.

Special Values

0x0000	Drawing default
0x0007	White
0x0008	Black
0xFF00	Drawing default
0xFF0x	Color indexes 0x000n, where n is in the range 1 through 7.
0xFF08	Color index 0 (reset color).

Set Background Color/Push and Set Background Color - Parameters

LEN (GLENGTH1)
Length of following data.

0x02 Only valid value.

COLOR (GBIT16)
Color-table index: Except for the special values, the values 0x0000 through 0xnnnn are allowed color indexes; that is, as many values as are allowed by the size of the LCT.

Special Values

0x0000
Drawing default
0x0007
White
0x0008
Black
0xFF00
Drawing default
0xFF0x
Color indexes 0x000n, where n is in the range 1 through 7.
0xFF08
Color index 0 (reset color).

Set Background Color/Push and Set Background Color - Topics

Select an item:
Syntax
Parameters
Glossary

Set Background Indexed Color/Push and Set Background Indexed Color

Set Background Indexed Color/Push and Set Background Indexed Color - Syntax

These orders set, or push and set, the value of the current background color attribute.

Set Background Indexed Color (GSBICOL)
X'A7' (LEN, FLAGS, INDEX)

Push and Set Background Indexed Color (GPSBICOL)
X'E7' (LEN, FLAGS, INDEX)

GSBICOL/GPSBICOL Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x04 Only valid value.

GSBICOL/GPSBICOL Parameter - FLAGS

FLAGS
Values:

DEFAULT (GBIT1)
Options:

0 Use specified *INDEX*
1 Use drawing default color.

SPEC (GBIT1)
Options:

0 Use index directly
1 Special value.

RES (GBIT6)
Reserved value, must be 0.

GSBICOL/GPSBICOL Parameter - INDEX

INDEX (GINDEX3)
Value for color index.

The value is a direct index into the current color table or a special value. The special values are:

1
Black
2
White
4
All ones
5
All zeros.

Set Background Indexed Color/Push and Set Background Indexed Color - Parameters

LEN (GLENGTH1)
Length of following data.

0x04 Only valid value.

FLAGS
Values:

DEFAULT (GBIT1)
Options:

0 Use specified *INDEX*
1 Use drawing default color.

SPEC (GBIT1)
Options:

0 Use index directly
1 Special value.

RES (GBIT6)
Reserved value, must be 0.

INDEX (GINDEX3)
Value for color index.

The value is a direct index into the current color table or a special value. The special values are:

1
Black
2
White
4
All ones
5
All zeros.

Set Background Indexed Color/Push and Set Background Indexed Color - Remarks

The value of the current background color attribute is pushed on to the stack by the Push and Set order only. Both orders then set the current background color attribute to the value specified in the order.

Set Background Indexed Color/Push and Set Background Indexed Color - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Background Mix/Push and Set Background Mix

Set Background Mix/Push and Set Background Mix - Syntax

These orders set, or push and set, the value of the current background mix attribute.

Set Background Mix (GSBMX)
X'0D' (MODE)

Push and Set Background Mix (GPSBMX)
X'4D' (MODE)

GSBMX/GPSBMX Parameter - MODE

MODE (GBIT8)
Mix-mode value:

0x00	Drawing default
0x01	OR
0x02	Overpaint
0x03	Reserved
0x04	Exclusive-OR
0x05	Leave Alone
0x06	AND
0x07	Subtract
0x08	Source AND (inverse destination)
0x09	All zeros
0x0A	Inverse (source OR destination)
0x0B	Inverse (source XOR destination)
0x0C	Inverse destination
0x0D	Source OR (inverse destination)
0x0E	Inverse source
0x0F	(Inverse source) OR destination
0x10	Inverse (source AND destination)
0x11	All ones.

Set Background Mix/Push and Set Background Mix - Parameters

MODE (GBIT8)

Mix-mode value:

0x00	Drawing default
0x01	OR
0x02	Overpaint
0x03	Reserved
0x04	Exclusive-OR
0x05	Leave Alone
0x06	AND
0x07	Subtract
0x08	Source AND (inverse destination)
0x09	All zeros
0x0A	Inverse (source OR destination)
0x0B	Inverse (source XOR destination)
0x0C	Inverse destination
0x0D	Source OR (inverse destination)
0x0E	Inverse source
0x0F	(Inverse source) OR destination
0x10	Inverse (source AND destination)
0x11	All ones.

Set Background Mix/Push and Set Background Mix - Remarks

The value of the current background mix attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the current background mix attribute to the value specified in the order.

Set Background Mix/Push and Set Background Mix - Topics

Select an item:

Syntax
Parameters
Remarks
Glossary

Set Character Angle/Push and Set Character Angle

Set Character Angle/Push and Set Character Angle - Syntax

These orders set, or push and set, the value of the current character angle attribute.

Set Character Angle (GSCA)
X'34' (LEN, AX, AY)

Push and Set Character Angle (GPSCA)
X'74' (LEN, AX, AY)

GSCA/GPSCA Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSCA/GPSCA Parameter - AX

AX (GROSOL)
X coordinate of point.

This point defines the angle of the character string.

GSCA/GPSCA Parameter - AY

AY (GROSOL)
Y coordinate of point.

This point defines the angle of the character string.

Set Character Angle/Push and Set Character Angle - Parameters

LEN (GLENGTH1)
Length of following data.

AX (GROSOL)
X coordinate of point.

This point defines the angle of the character string.

AY (GROSOL)
Y coordinate of point.

This point defines the angle of the character string.

Set Character Angle/Push and Set Character Angle - Remarks

The value of the current character angle attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current character angle to the value specified in the order.

Set Character Angle/Push and Set Character Angle - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Character Break Extra/Push and Set Character Break Extra

Set Character Break Extra/Push and Set Character Break Extra - Syntax

These orders set, or push and set, the value of the current character break extra attribute.

Set Character Break Extra (GSCBE)
X'05' (LEN, FLAGS, RES2, INC)

Push and Set Character Break Extra (GPSCBE)
X'45' (LEN, FLAGS, RES2, INC)

GSCBE/GPSCBE Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSCBE/GPSCBE Parameter - FLAGS

FLAGS
Values as follows:

DEFAULT (GBIT1)
Values as follows:

B"0"	Set to specified value.
B"1"	Set to drawing default.

RES1 (GBIT7)
Reserved value, must be 0.

GSCBE/GPSCBE Parameter - RES2

RES2 (GUNDF1)
Reserved value, must be 0.

GSCBE/GPSCBE Parameter - INC

INC (GROF)
Increment.

Set Character Break Extra/Push and Set Character Break Extra - Parameters

LEN (GLENGTH1)
Length of following data.

FLAGS
Values as follows:

DEFAULT (GBIT1)
Values as follows:

B"0"	Set to specified value.
B"1"	Set to drawing default.

RES1 (GBIT7)
Reserved value, must be 0.

RES2 (GUNDF1)
Reserved value, must be 0.

INC (GROF)
Increment.

Set Character Break Extra/Push and Set Character Break Extra - Remarks

The value of the current character break extra attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current character break extra attribute to the value specified in the order.

Set Character Break Extra/Push and Set Character Break Extra - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Character Cell/Push and Set Character Cell

Set Character Cell/Push and Set Character Cell - Syntax

These orders set, or push and set, the value of the current character cell-size attribute.

Set Character Cell (GSCC)
X'33' (LEN, CELLX, CELLY, CELLXF, CELLYF, FLAGS, RES)

Push and Set Character Cell (GPSCC)
X'03' (LEN, CELLX, CELLY, CELLXF, CELLYF, FLAGS, RES)

GSCC/GPSCC Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSCC/GPSCC Parameter - CELLX

CELLX (GROSOL)
X part of character cell-size attribute.

GSCC/GPSCC Parameter - CELLY

CELLY (GROSOL)
Y part of character cell-size attribute.

GSCC/GPSCC Parameter - CELLXF

CELLXF (GUSHORT)
Fractional X part of character cell-size attribute.

This parameter is optional.

GSCC/GPSCC Parameter - CELLYF

CELLYF (GUSHORT)
Fractional Y part of character cell-size attribute.

This parameter must be present if *CELLXF* parameter is present.

GSCC/GPSCC Parameter - FLAGS

FLAGS
Internal flags.

This parameter is optional.

NOTDEFLT (GBIT1)
Values:

0	A cell size of zero sets drawing default
1	A cell size of zero sets to zero.

RES (GBIT7)
Reserved.

0000000	Only valid value.

GSCC/GPSCC Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

This parameter must be present if *FLAGS* parameter is present.

Set Character Cell/Push and Set Character Cell - Parameters

LEN (GLENGTH1)
Length of following data.

CELLX (GROSOL)
X part of character cell-size attribute.

CELLY (GROSOL)
Y part of character cell-size attribute.

CELLXF (GUSHORT)
Fractional X part of character cell-size attribute.

This parameter is optional.

CELLYF (GUSHORT)
Fractional Y part of character cell-size attribute.

This parameter must be present if *CELLXF* parameter is present.

FLAGS
Internal flags.

This parameter is optional.

NOTDEFLT (GBIT1)
Values:

0	A cell size of zero sets drawing default
1	A cell size of zero sets to zero.

RES (GBIT7)
Reserved.

0000000	Only valid value.

RES (GBIT8)
Reserved value, must be 0.

This parameter must be present if *FLAGS* parameter is present.

Set Character Cell/Push and Set Character Cell - Remarks

The value of the current character cell-size attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current character cell-size attribute to the value in the order.

Set Character Cell/Push and Set Character Cell - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Character Direction/Push and Set Character Direction

Set Character Direction/Push and Set Character Direction - Syntax

These orders set, or push and set, the value of the current character direction attribute.

Set Character Direction (GSCD)
X'3A' (DIRECTION)

Push and Set Character Direction (GPSCD)
X'7A' (DIRECTION)

GSCD/GPSCD Parameter - DIRECTION

DIRECTION (GBIT8)
Value for character direction:

All other values are reserved.

0x00	Drawing default
0x01	Left to right
0x02	Top to bottom
0x03	Right to left
0x04	Bottom to top.

Set Character Direction/Push and Set Character Direction - Parameters

DIRECTION (GBIT8)
Value for character direction:

All other values are reserved.

0x00	Drawing default
0x01	Left to right
0x02	Top to bottom
0x03	Right to left
0x04	Bottom to top.

Set Character Direction/Push and Set Character Direction - Remarks

The value of the current character direction attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current character direction attribute to the value in the order.

Set Character Direction/Push and Set Character Direction - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Character Extra/Push and Set Character Extra

Set Character Extra/Push and Set Character Extra - Syntax

These orders set, or push and set, the value of the current character extra attribute.

Set Character Extra (GSCE)

X'17' (LEN, FLAGS, RES2, INC)

Push and Set Character Extra (GPSCE)
X'57' (LEN, FLAGS, RES2, INC)

GSCE/GPSCE Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSCE/GPSCE Parameter - FLAGS

FLAGS
Values as follows:

DEFAULT (GBIT1)
Values as follows:

B"0"	Set to specified value.
B"1"	Set to drawing default.

RES1 (GBIT7)
Reserved value, must be 0.

GSCE/GPSCE Parameter - RES2

RES2 (GUNDF1)
Reserved value, must be 0.

GSCE/GPSCE Parameter - INC

INC (GROF)
Increment.

Set Character Extra/Push and Set Character Extra - Parameters

LEN (GLENGTH1)
Length of following data.

FLAGS
Values as follows:

DEFAULT (GBIT1)
Values as follows:

B"0"	Set to specified value.
B"1"	Set to drawing default.

RES1 (GBIT7)
Reserved value, must be 0.

RES2 (GUNDF1)
Reserved value, must be 0.

INC (GROF)
Increment.

Set Character Extra/Push and Set Character Extra - Remarks

The value of the current character extra attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders set the value of the current character extra attribute to the value specified in the order.

Set Character Extra/Push and Set Character Extra - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Character Precision/Push and Set Character Precision

Set Character Precision/Push and Set Character Precision - Syntax

These orders set, or push and set, the value of the current character precision attribute.

Set Character Precision (GSCR)
X'39' (PREC)

Push and Set Character Precision (GPSCR)
X'79' (PREC)

GSCR/GPSCR Parameter - PREC

PREC (GBIT8)
Value for character-precision attribute:

All other values are reserved.

0x00	Drawing default
0x01	String precision
0x02	Character precision
0x03	Stroke precision

Set Character Precision/Push and Set Character Precision - Parameters

PREC (GBIT8)
Value for character-precision attribute:

All other values are reserved.

0x00	Drawing default
0x01	String precision
0x02	Character precision
0x03	Stroke precision

Set Character Precision/Push and Set Character Precision - Remarks

The value of the current character precision attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current character precision attribute to the value in the order.

Set Character Precision/Push and Set Character Precision - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Character Set/Push and Set Character Set

Set Character Set/Push and Set Character Set - Syntax

These orders set, or push and set, the value of the current character-set attribute.

Set Character Set (GSCS)
X'38' (LCID)

Push and Set Character Set (GPSCS)
X'78' (LCID)

GSCS/GPSCS Parameter - LCID

LCID (GUCHAR)
Local identifier (LCID) for the character set:

0x00	Drawing default
0x01-0xFE	Lcid for the symbol set
0xFF	Special character set.

Set Character Set/Push and Set Character Set - Parameters

LCID (GUCHAR)
Local identifier (LCID) for the character set:

0x00	Drawing default
0x01-0xFE	Lcid for the symbol set
0xFF	Special character set.

Set Character Set/Push and Set Character Set - Remarks

The value of the current character-set attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current character-set attribute to the value in the order.

Set Character Set/Push and Set Character Set - Topics

Select an item:
Syntax
Parameters

Remarks
Glossary

Set Character Shear/Push and Set Character Shear

Set Character Shear/Push and Set Character Shear - Syntax

These orders set, or push and set, the value of the current character shear attribute.

Set Character Shear (GSCH)
X'35' (LEN, HX, HY)

Push and Set Character Shear (GPSCH)
X'75' (LEN, HX, HY)

GSCH/GPSCH Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSCH/GPSCH Parameter - HX

HX (GROSOL)
Dividend of shear ratio.

GSCH/GPSCH Parameter - HY

HY (GROSOL)
Divisor of shear ratio.

Set Character Shear/Push and Set Character Shear - Parameters

LEN (GLENGTH1)
Length of following data.

HX (GROSOL)
Dividend of shear ratio.

HY (GROSOL)
Divisor of shear ratio.

Set Character Shear/Push and Set Character Shear - Remarks

When *HX* and *HY* are both 0, the drawing default is set. The value of the current character shear attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current character shear attribute to the value in the order.

Set Character Shear/Push and Set Character Shear - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Clip Path

Set Clip Path - Syntax

This order sets the current clip path.

Set Clip Path (GSCPTH)
X'B4' (LEN, FLAGS, RES, PTHID)

GSCPTH Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSCPTH Parameter - FLAGS

FLAGS
Extra functions:

RES (GBIT1)
Reserved value, must be 0.

FILL (GBIT1)
Values:

0	Alternate mode
1	Winding mode.

INTER (GBIT1)
Values:

0	Set to specified path
1	Set to intersection of specified and current clip path.

RES2 (GBIT5)
Reserved value, must be 0.

GSCPTH Parameter - RES

RES (GBIT8)
Reserved.

0x00	Only valid value.

GSCPTH Parameter - PTHID

PTHID (GLONG)
Path identifier.

0x00000000	No clipping.
0x00000001-0xFFFFFFFF	Path identifier.

Set Clip Path - Parameters

LEN (GLENGTH1)
Length of following data.

FLAGS
Extra functions:

RES (GBIT1)
Reserved value, must be 0.

FILL (GBIT1)
Values:

0	Alternate mode
1	Winding mode.

INTER (GBIT1)
Values:

0	Set to specified path
1	Set to intersection of specified and current clip path.

RES2 (GBIT5)
Reserved value, must be 0.

RES (GBIT8)
Reserved.

0x00	Only valid value.

PTHID (GLONG)
Path identifier.

0x00000000	No clipping.
0x00000001-0xFFFFFFFF	Path identifier.

Set Clip Path - Topics

Select an item:
Syntax
Parameters
Glossary

Set Color/Push and Set Color

Set Color/Push and Set Color - Syntax

These orders set, or push and set, the value of the current color attribute.

Set Color (GSCOL)
X'0A' (COL)

Push and Set Color (GPSCOL)
X'4A' (COL)

GSCOL/GPSCOL Parameter - COL

COL (GBIT8)
Value for color attribute:

0x00-0x08	These one-byte values are converted to two-byte values by preceding the value with 0xFF. The resultant is then treated as a two-byte value as defined by the Set Extended Color/Push and Set Extended Color order.
Other	Reserved values.

Set Color/Push and Set Color - Parameters

COL (GBIT8)
Value for color attribute:

0x00-0x08	These one-byte values are converted to two-byte values by preceding the value with 0xFF. The resultant is then treated as a two-byte value as defined by the Set Extended Color/Push and Set Extended Color order.
Other	Reserved values.

Set Color/Push and Set Color - Remarks

The value of the current color attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current color attribute to the value in the order.

Set Color/Push and Set Color - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Current Position/Push and Set Current Position

Set Current Position/Push and Set Current Position - Syntax

These orders set, or push and set, the value of the current position.

Set Current Position (GSCP)
X'21' (LEN, P)

Push and Set Current Position (GPSCP)
X'61' (LEN, P)

GSCP/GPSCP Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSCP/GPSCP Parameter - P

P (GPOINT)
Coordinate data.

Set Current Position/Push and Set Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P (GPOINT)
Coordinate data.

Set Current Position/Push and Set Current Position - Remarks

The value of the current position is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current position to the value in the order.

Set Current Position/Push and Set Current Position - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Extended Color/Push and Set Extended Color

Set Extended Color/Push and Set Extended Color - Syntax

These orders set, or push and set, the value of the current color attribute.

Set Extended Color (GSECOL)
X'26' (LEN, COLOR)

Push and Set Extended Color (GPSECOL)
X'66' (LEN, COLOR)

GSECOL/GPSECOL Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x02	Only valid value.

GSECOL/GPSECOL Parameter - COLOR

COLOR (GBIT16)
Color-table index. Except for the special values, the values 0x0000 through 0xnnnn are allowed color indexes; that is, as many values as are allowed by the size of the LCT.

Special Values

0x0000	Drawing default
0x0007	White
0x0008	Black
0xFF00	Drawing default
0xFF0n	Color indexes 0x000n, where n is in the range 1 through 7.
0xFF08	Color index 0 (reset color).

Set Extended Color/Push and Set Extended Color -

Parameters

LEN (GLENGTH1)
Length of following data.

0x02	Only valid value.

COLOR (GBIT16)
Color-table index. Except for the special values, the values 0x0000 through 0xnnnn are allowed color indexes; that is, as many values as are allowed by the size of the LCT.

Special Values

0x0000	Drawing default
0x0007	White
0x0008	Black
0xFF00	Drawing default
0xFF0n	Color indexes 0x000n, where n is in the range 1 through 7.
0xFF08	Color index 0 (reset color).

Set Extended Color/Push and Set Extended Color - Remarks

The value of the current extended color attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current extended color attribute to the value in the order.

Set Extended Color/Push and Set Extended Color - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Fractional Line Width/Push and Set Fractional Line Width

Set Fractional Line Width/Push and Set Fractional Line Width - Syntax

These orders set, or push and set, the value of the current line-width attribute.

Set Fractional Line Width (GSFLW)
X'11' (LEN, LINEWIDTH)

Push and Set Fractional Line Width (GPSFLW)
X'51' (LEN, LINEWIDTH)

GSFLW/GPSFLW Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x02	Only valid value.

GSFLW/GPSFLW Parameter - LINEWIDTH

LINEWIDTH (GROUFS)
Value for the line-width attribute.

The nonzero value is an integral and fractional multiplier of the normal line width:

0x0000	Drawing default
0x0001-0xFFFF	Multiplier of normal line width.

Set Fractional Line Width/Push and Set Fractional Line Width - Parameters

LEN (GLENGTH1)
Length of following data.

0x02	Only valid value.

LINEWIDTH (GROUFS)
Value for the line-width attribute.

The nonzero value is an integral and fractional multiplier of the normal line width:

0x0000	Drawing default
0x0001-0xFFFF	Multiplier of normal line width.

Set Fractional Line Width/Push and Set Fractional Line Width

- Remarks

The value of the current line-width attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current line-width attribute to the value in the order.

Set Fractional Line Width/Push and Set Fractional Line Width - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Indexed Color/Push and Set Indexed Color

Set Indexed Color/Push and Set Indexed Color - Syntax

These orders set, or push and set, the value of the current color attribute.

Set Indexed Color (GSICOL)
X'A6' (LEN, FLAGS, INDEX)

Push and Set Indexed Color (GPSICOL)
X'E6' (LEN, FLAGS, INDEX)

GSICOL/GPSICOL Parameter - LEN

LEN (GLENGTH1)
Length of following data.

0x04 Only valid value.

GSICOL/GPSICOL Parameter - FLAGS

FLAGS
Values:

DEFAULT (GBIT1)
Options:

0	Use specified index
1	Use drawing default color.

SPEC (GBIT1)
Options:

0	Use index directly
1	Special value.

RES (GBIT6)
Reserved value, must be 0.

GSICOL/GPSICOL Parameter - INDEX

INDEX (GINDEX3)
Value for color index.

The value is a direct index into the current color table or a special value.

The table can be the standard table, or one loaded by the user. The special values are:

1
Black
2
White
4
All ones
5
All zeros.

Set Indexed Color/Push and Set Indexed Color - Parameters

LEN (GLENGTH1)
Length of following data.

0x04 Only valid value.

FLAGS
Values:

DEFAULT (GBIT1)
Options:

0	Use specified index
1	Use drawing default color.

SPEC (GBIT1)
Options:

0	Use index directly
1	Special value.

RES (GBIT6)
Reserved value, must be 0.

INDEX (GINDEX3)
Value for color index.

The value is a direct index into the current color table or a special value.

The table can be the standard table, or one loaded by the user. The special values are:

1
Black
2
White
4
All ones
5
All zeros.

Set Indexed Color/Push and Set Indexed Color - Remarks

The value of the current color attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current color attribute to the value in the order.

Set Indexed Color/Push and Set Indexed Color - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Individual Attribute/Push and Set Individual Attribute

Set Individual Attribute/Push and Set Individual Attribute - Syntax

These orders set, or push and set, the value of the color, background color, mix, or background mix attribute for the line character, marker, pattern, or image primitive type.

Set Individual Attribute (GSIA)
X'14' (LEN, ATYPE, PTYPE, FLAG1, VAL)

Push and Set Individual Attribute (GPSIA)
X'54' (LEN, ATYPE, PTYPE, FLAG1, VAL)

GSIA/GPSIA Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSIA/GPSIA Parameter - ATYPE

ATYPE (GBIT8)
Attribute type:

0x1	Color
0x2	Background color
0x3	Mix
0x4	Background Mix
Other	All other values are reserved.

GSIA/GPSIA Parameter - PTYPE

PTYPE (GBIT8)
Primitive type:

0x1	Line
0x2	Character
0x3	Marker
0x4	Pattern
0x5	Image
Other	All other values are reserved.

GSIA/GPSIA Parameter - FLAG1

FLAG1
Values:

DEFAULT (GBIT1)
Options:

0	Use specified value
1	Use drawing default color.

SPEC (GBIT1)
Options:

0
Use value directly
1
Special Value.

RES (GBIT6)
Reserved value, must be 0.

GSIA/GPSIA Parameter - VAL

VAL (GINDATT)
Color index value.

For colors, the value is a direct index into the current color table or a special value.

The table can be the standard table, or one loaded by the user. The special values are:

1
Black
2
White
4
All ones
5
All zeros.

Set Individual Attribute/Push and Set Individual Attribute - Parameters

LEN (GLENGTH1)
Length of following data.

ATYPE (GBIT8)
Attribute type:

0x1	Color
0x2	Background color
0x3	Mix
0x4	Background Mix
Other	All other values are reserved.

PTYPE (GBIT8)
Primitive type:

0x1
Line
0x2
Character
0x3
Marker
0x4
Pattern
0x5
Image

Other
All other values are reserved.

FLAG1
Values:

DEFAULT (GBIT1)
Options:

0
Use specified value
1
Use drawing default color.

SPEC (GBIT1)
Options:

0
Use value directly
1
Special Value.

RES (GBIT6)
Reserved value, must be 0.

VAL (GINDATT)
Color index value.

For colors, the value is a direct index into the current color table or a special value.

The table can be the standard table, or one loaded by the user. The special values are:

1
Black
2
White
4
All ones
5
All zeros.

Set Individual Attribute/Push and Set Individual Attribute - Remarks

The value of the current attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the individual attribute to the value in the order.

Set Individual Attribute/Push and Set Individual Attribute - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Line End/Push and Set Line End

Set Line End/Push and Set Line End - Syntax

These orders set, or push and set, the value of the current line-end attribute.

Set Line End (GSLE)
X'1A' (LINEEND)

Push and Set Line End (GPSLE)
X'5A' (LINEEND)

GSLE/GPSLE Parameter - LINEEND

LINEEND (GBIT8)
Value for the line-end attribute:

0x00	Drawing default
0x01	Flat
0x02	Square
0x03	Round
Other	Reserved values.

Set Line End/Push and Set Line End - Parameters

LINEEND (GBIT8)
Value for the line-end attribute:

0x00	Drawing default
0x01	Flat
0x02	Square
0x03	Round
Other	Reserved values.

Set Line End/Push and Set Line End - Remarks

The value of the current line-end attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current line-end attribute to the value in the order.

Set Line End/Push and Set Line End - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Line Join/Push and Set Line Join

Set Line Join/Push and Set Line Join - Syntax

These orders set the value of the current line-join attribute.

Set Line Join (GSLJ)
X'1B' (LINEJOIN)

Push and Set Line Join (GPSLJ)
X'5B' (LINEJOIN)

GSLJ/GPSLJ Parameter - LINEJOIN

LINEJOIN (GBIT8)
Value for line-join attribute:

0x00	Drawing default
0x01	Bevel
0x02	Round
0x03	Miter
Other	Reserved values.

Set Line Join/Push and Set Line Join - Parameters

LINEJOIN (GBIT8)
Value for line-join attribute:

0x00	Drawing default
0x01	Bevel
0x02	Round
0x03	Miter
Other	Reserved values.

Set Line Join/Push and Set Line Join - Remarks

The value of the current line-join attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current line-join attribute to the value in the order.

Set Line Join/Push and Set Line Join - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Line Type/Push and Set Line Type

Set Line Type/Push and Set Line Type - Syntax

These orders set, or push and set, the value of the current line-type attribute.

Set Line Type (GSLT)
X'18' (LINETYPE)

Push and Set Line Type (GPSLT)
X'58' (LINETYPE)

GSLT/GPSLT Parameter - LINETYPE

LINETYPE (GBIT8)
Value for line-type attribute.

The value is an index into a notational line-type table:

0x00	Drawing default
0x01	Dotted line
0x02	Short dashed line
0x03	Dash-dot line
0x04	Double dotted line
0x05	Long dashed line
0x06	Dash-double-dot line
0x07	Solid line
0x08	Invisible line
Other	Reserved values.

Set Line Type/Push and Set Line Type - Parameters

LINETYPE (GBIT8)
Value for line-type attribute.

The value is an index into a notational line-type table:

0x00	Drawing default
0x01	Dotted line
0x02	Short dashed line
0x03	Dash-dot line
0x04	Double dotted line
0x05	Long dashed line
0x06	Dash-double-dot line
0x07	Solid line
0x08	Invisible line
Other	Reserved values.

Set Line Type/Push and Set Line Type - Remarks

The value of the current line-type attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then set the value of the current line-type attribute to the value in the order.

Set Line Type/Push and Set Line Type - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Line Width/Push and Set Line Width

Set Line Width/Push and Set Line Width - Syntax

These orders set, or push and set, the value of the current line-width attribute to the value specified in the order.

Set Line Width (GSLW)
X'19' (LINEWIDTH)

Push and Set Line Width (GPSLW)
X'59' (LINEWIDTH)

GSLW/GPSLW Parameter - LINEWIDTH

LINEWIDTH (GBIT8)
Value for line-width attribute:

0x00	Drawing default
0x01-0xFF	Integral multiplier of normal line width.

Set Line Width/Push and Set Line Width - Parameters

LINEWIDTH (GBIT8)
Value for line-width attribute:

0x00	Drawing default
0x01-0xFF	Integral multiplier of normal line width.

Set Line Width/Push and Set Line Width - Remarks

The value of the current line-width attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current line-width attribute to the value in the order.

Set Line Width/Push and Set Line Width - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Marker Cell/Push and Set Marker Cell

Set Marker Cell/Push and Set Marker Cell - Syntax

These orders set, or push and set, the value of the current marker cell-size attribute.

Set Marker Cell (GSMC)
X'37' (LEN, CELLX, CELLY, FLAGS, RES)

Push and Set Marker Cell (GPSMC)

X'77' (LEN, CELLX, CELLY, FLAGS, RES)

GSMC/GPSMC Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSMC/GPSMC Parameter - CELLX

CELLX (GROSOL)
X part of marker cell-size attribute.

GSMC/GPSMC Parameter - CELLY

CELLY (GROSOL)
Y part of marker cell-size attribute.

GSMC/GPSMC Parameter - FLAGS

FLAGS
This is an optional extension.

Values:

NOTDEFLT (GBIT1)
Options:

0	A cell size of zero sets drawing default
1	A cell size of zero sets to zero.

RES (GBIT7)
Reserved value, must be 0.

GSMC/GPSMC Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

Set Marker Cell/Push and Set Marker Cell - Parameters

LEN (GLENGTH1)
Length of following data.

CELLX (GROSOL)
X part of marker cell-size attribute.

CELLY (GROSOL)
Y part of marker cell-size attribute.

FLAGS
This is an optional extension.

Values:

NOTDEFLT (GBIT1)
Options:

0	A cell size of zero sets drawing default
1	A cell size of zero sets to zero.

RES (GBIT7)
Reserved value, must be 0.

RES (GBIT8)
Reserved value, must be 0.

Set Marker Cell/Push and Set Marker Cell - Remarks

The value of the current marker cell-size attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current marker cell-size attribute to the value in the order.

Set Marker Cell/Push and Set Marker Cell - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Marker Precision/Push and Set Marker Precision

Set Marker Precision/Push and Set Marker Precision - Syntax

These orders set, or push and set, the value of the current marker-precision attribute.

Set Marker Precision (GSMP)
X'3B' (PREC)

Push and Set Marker Precision (GPSMP)
X'7B' (PREC)

GSMP/GPSMP Parameter - PREC

PREC (GBIT8)
Value for marker-precision attribute:

0x00	Drawing default
0x01	String precision
0x02	Character precision
0x03	Stroke precision
Other	Reserved values.

Set Marker Precision/Push and Set Marker Precision - Parameters

PREC (GBIT8)
Value for marker-precision attribute:

0x00	Drawing default
0x01	String precision
0x02	Character precision
0x03	Stroke precision
Other	Reserved values.

Set Marker Precision/Push and Set Marker Precision - Remarks

The value of the current marker-precision attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current-marker precision attribute to the value in the order.

Set Marker Precision/Push and Set Marker Precision - Topics

Select an item:

Syntax
Parameters
Remarks
Glossary

Set Marker Set/Push and Set Marker Set

Set Marker Set/Push and Set Marker Set - Syntax

These orders set, or push and set, the value of the current marker symbol-set attribute.

Set Marker Set (GSMS)
X'3C' (LCID)

Push and Set Marker Set (GPSMS)
X'7C' (LCID)

GSMS/GPSMS Parameter - LCID

LCID (GUCHAR)
Local identifier (LCID) for the marker set:

0x00	Drawing default
0x01-0xFE	LCID for the coded font
0xFF	Special marker set.

Set Marker Set/Push and Set Marker Set - Parameters

LCID (GUCHAR)
Local identifier (LCID) for the marker set:

0x00	Drawing default
0x01-0xFE	LCID for the coded font
0xFF	Special marker set.

Set Marker Set/Push and Set Marker Set - Remarks

The value of the current marker symbol-set attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current marker symbol-set attribute to the value in the order.

Set Marker Set/Push and Set Marker Set - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Marker Symbol/Push and Set Marker Symbol

Set Marker Symbol/Push and Set Marker Symbol - Syntax

These orders set, or push and set, the value of the current marker symbol attribute.

Set Marker Symbol (GSMT)
X'29' (N)

Push and Set Marker Symbol (GPSMT)
X'69' (N)

GSMT/GPSMT Parameter - N

N (GBIT8)
Value of marker symbol code point. **Special marker set**

When this is selected (**lcid** = 0xFF), the values are:

0x00
Drawing default
0x01
Cross
0x02
Plus
0x03
Diamond
0x04
Square
0x05
6-point star
0x06
8-point star
0x07
Filled diamond
0x08
Filled square
0x09
Dot
0x0A

Small circle

0x40

Blank

Other

Reserved values.

Marker set

Values are as follows for any other set:

0x00

Drawing default

0x01-0xFF

These are the code points into the current marker set.

Set Marker Symbol/Push and Set Marker Symbol - Parameters

N (GBIT8)

Value of marker symbol code point. **Special marker set**

When this is selected (**lcid** = 0xFF), the values are:

0x00

Drawing default

0x01

Cross

0x02

Plus

0x03

Diamond

0x04

Square

0x05

6-point star

0x06

8-point star

0x07

Filled diamond

0x08

Filled square

0x09

Dot

0x0A

Small circle

0x40

Blank

Other

Reserved values.

Marker set

Values are as follows for any other set:

0x00

Drawing default

0x01-0xFF

These are the code points into the current marker set.

Set Marker Symbol/Push and Set Marker Symbol - Remarks

The value of the current marker symbol attribute is pushed on to the Segment Call Stack by the Push and Set order only. Both orders then

set the value of the current marker symbol attribute to the value in the order.

Set Marker Symbol/Push and Set Marker Symbol - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Mix/Push and Set Mix

Set Mix/Push and Set Mix - Syntax

These orders set, or push and set, the value of the current mix attribute.

Set Mix (GSMX)
X'0C' (MODE)

Push and Set Mix (GPSMX)
X'4C' (MODE)

GSMX/GPSMX Parameter - MODE

MODE (GBIT8)
Mix-mode value:

0x00	Drawing default
0x01	OR
0x02	Overpaint
0x03	Reserved
0x04	Exclusive-OR
0x05	Leave alone
0x06	AND
0x07	Subtract
0x08	Source AND (inverse destination)
0x09	All zeros
0x0A	Inverse (source OR destination)
0x0B	Inverse (source XOR destination)
0x0C	Inverse destination
0x0D	Source OR (inverse destination)
0x0E	Inverse source
0x0F	(Inverse source) OR destination
0x10	Inverse (source AND destination)
0x11	All ones.
Other	Reserved values.

Set Mix/Push and Set Mix - Parameters

MODE (GBIT8)
Mix-mode value:

0x00	Drawing default
0x01	OR
0x02	Overpaint
0x03	Reserved
0x04	Exclusive-OR
0x05	Leave alone
0x06	AND
0x07	Subtract
0x08	Source AND (inverse destination)
0x09	All zeros
0x0A	Inverse (source OR destination)
0x0B	Inverse (source XOR destination)
0x0C	Inverse destination
0x0D	Source OR (inverse destination)
0x0E	Inverse source
0x0F	(Inverse source) OR destination
0x10	Inverse (source AND destination)
0x11	All ones.
Other	Reserved values.

Set Mix/Push and Set Mix - Remarks

The value of the current mix attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current mix attribute to the value in the order.

Set Mix/Push and Set Mix - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Model Transform/Push and Set Model Transform

Set Model Transform/Push and Set Model Transform - Syntax

These orders set, or push and set, values in the current model transform.

Set Model Transform (GSTM)
X'24' (LEN, RES, FLAGS, MASK, MX[LEN])

Push and Set Model Transform (GPSTM)
X'64' (LEN, RES, FLAGS, MASK, MX[LEN])

GSTM/GPSTM Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSTM/GPSTM Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GSTM/GPSTM Parameter - FLAGS

FLAGS
Values:

RES (GBIT6)
Reserved value, must be 0.

CM (GBIT2)
Matrix control bits:

B"00"	Unity matrix
B"01"	Concatenate after
B"10"	Concatenate before
B"11"	Overwrite.

GSTM/GPSTM Parameter - MASK

MASK (GBIT16)
Load mask.

GSTM/GPSTM Parameter - MX[LEN]

MX[LEN] (GROSOL)
Matrix values.

The matrix size is based on the number of bits set in MASK.

Set Model Transform/Push and Set Model Transform - Parameters

LEN (GLENGTH1)
Length of following data.

RES (GBIT8)
Reserved value, must be 0.

FLAGS
Values:

RES (GBIT6)
Reserved value, must be 0.

CM (GBIT2)
Matrix control bits:

B"00"	Unity matrix
B"01"	Concatenate after
B"10"	Concatenate before
B"11"	Overwrite.

MASK (GBIT16)
Load mask.

MX[LEN] (GROSOL)
Matrix values.

The matrix size is based on the number of bits set in MASK.

Set Model Transform/Push and Set Model Transform - Remarks

The value of the current model transform is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set values in the current model transform as specified in the order.

Set Model Transform/Push and Set Model Transform - Topics

Select an item:
Syntax
Parameters
Remarks

Glossary

Set Pattern Reference Point/Push and Set Pattern Reference Point

Set Pattern Reference Point/Push and Set Pattern Reference Point - Syntax

These orders set, or push and set, the value of the current pattern reference-point attribute.

Set Pattern Reference Point (GSPRP)
X'A0' (LEN, FLAGS, RES, PREF)

Push and Set Pattern Reference Point (GPSPRP)
X'E0' (LEN, FLAGS, RES, PREF)

GSPRP/GPSPRP Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSPRP/GPSPRP Parameter - FLAGS

FLAGS
Values:

DEFAULT (GBIT1)
Options:

0	Set to specified value
1	Set to the drawing default.

RES (GBIT7)
Reserved value, must be 0.

GSPRP/GPSPRP Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GSPRP/GPSPRP Parameter - PREF

PREF (GPOINT)
Coordinate data of the pattern-reference point.

Set Pattern Reference Point/Push and Set Pattern Reference Point - Parameters

LEN (GLENGTH1)
Length of following data.

FLAGS
Values:

DEFAULT (GBIT1)
Options:

0	Set to specified value
1	Set to the drawing default.

RES (GBIT7)
Reserved value, must be 0.

RES (GBIT8)
Reserved value, must be 0.

PREF (GPOINT)
Coordinate data of the pattern-reference point.

Set Pattern Reference Point/Push and Set Pattern Reference Point - Remarks

The value of the current pattern reference-point attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current reference-point attribute to the value in the order.

Set Pattern Reference Point/Push and Set Pattern Reference Point - Topics

Select an item:
Syntax

Parameters
Remarks
Glossary

Set Pattern Set/Push and Set Pattern Set

Set Pattern Set/Push and Set Pattern Set - Syntax

These orders set, or push and set, the value of the current pattern symbol-set attribute.

Set Pattern Set (GSPS)
X'08' (LCID)

Push and Set Pattern Set (GPSPS)
X'48' (LCID)

GSPS/GPSPS Parameter - LCID

LCID (GUCHAR)
Local identifier (LCID) for the pattern set:

0x00	Drawing default
0x01-0xFE	LCID for the symbol set
0xFF	Special pattern set.

Set Pattern Set/Push and Set Pattern Set - Parameters

LCID (GUCHAR)
Local identifier (LCID) for the pattern set:

0x00	Drawing default
0x01-0xFE	LCID for the symbol set
0xFF	Special pattern set.

Set Pattern Set/Push and Set Pattern Set - Remarks

The value of the current pattern symbol-set attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current pattern symbol-set attribute to the value in the order.

Set Pattern Set/Push and Set Pattern Set - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Pattern Symbol/Push and Set Pattern Symbol

Set Pattern Symbol/Push and Set Pattern Symbol - Syntax

These orders set, or push and set, the value of the current pattern-symbol attribute.

Set Pattern Symbol (GSPT)
X'28' (PATT)

Push and Set Pattern Symbol (GPSPT)
X'09' (PATT)

GSPT/GPSPT Parameter - PATT

PATT (GBIT8)
Value for pattern-symbol attribute.

Special pattern set

When this is selected (**lcid** = 0xFF), the values are:

0x00	Drawing default
0x01-0x08	Density one through density eight (decreasing)
0x09	Vertical lines
0x0A	Horizontal lines
0x0B	Diagonal lines 1 (bottom-left to top-right)
0x0C	Diagonal lines 2 (bottom-left to top-right)
0x0D	Diagonal lines 1 (top-left to bottom-right)
0x0E	Diagonal lines 2 (top-left to bottom-right)
0x0F	No shading
0x10	

Solid shading

0x40
Blank.

Other
Reserved values.

Pattern set

Values are as follows for any other set:

0x00
Drawing default

0x01-0xFF
These are the code points into the current pattern set.

Set Pattern Symbol/Push and Set Pattern Symbol - Parameters

PATT (GBIT8)
Value for pattern-symbol attribute.

Special pattern set

When this is selected (**lcid** = 0xFF), the values are:

0x00
Drawing default

0x01-0x08
Density one through density eight (decreasing)

0x09
Vertical lines

0x0A
Horizontal lines

0x0B
Diagonal lines 1 (bottom-left to top-right)

0x0C
Diagonal lines 2 (bottom-left to top-right)

0x0D
Diagonal lines 1 (top-left to bottom-right)

0x0E
Diagonal lines 2 (top-left to bottom-right)

0x0F
No shading

0x10
Solid shading

0x40
Blank.

Other
Reserved values.

Pattern set

Values are as follows for any other set:

0x00
Drawing default

0x01-0xFF
These are the code points into the current pattern set.

Set Pattern Symbol/Push and Set Pattern Symbol - Remarks

The value of the current pattern-symbol attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current pattern-symbol attribute to the value in the order.

Set Pattern Symbol/Push and Set Pattern Symbol - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Pick Identifier/Push and Set Pick Identifier

Set Pick Identifier/Push and Set Pick Identifier - Syntax

These orders set, or push and set, the value of the current pick identifier.

Set Pick Identifier (GSPIK)
X'43' (LEN, PKID)

Push and Set Pick Identifier (GPSPIK)
X'23' (LEN, PKID)

GSPIK/GPSPIK Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSPIK/GPSPIK Parameter - PKID

PKID (GLONG)
Pick identifier.

Set Pick Identifier/Push and Set Pick Identifier - Parameters

LEN (GLENGTH1)
Length of following data.

PKID (GLONG)
Pick identifier.

Set Pick Identifier/Push and Set Pick Identifier - Remarks

The value of the current pick identifier is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the value of the current pick identifier to the value in the order.

Set Pick Identifier/Push and Set Pick Identifier - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Segment Boundary

Set Segment Boundary - Syntax

This order defines the maximum extent of the boundaries of the associated root segment. It is valid *only* in a root segment prolog.

Set Segment Boundary (GSSB)
X'32' (LEN, RES, MASK, BB[LEN])

GSSB Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSSB Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GSSB Parameter - MASK

MASK
Values:

RES1 (GBIT2)
Reserved value, must be 0.

XL (GBIT1)
X left limit.

0	Not included in list of *BB* values
1	Is included in list of *BB* values.

XR (GBIT1)
X right limit.

0	Not included in list of *BB* values
1	Is included in list of *BB* values.

YB (GBIT1)
Y bottom limit.

0	Not included in list of *BB* values
1	Is included in list of *BB* values.

YT (GBIT1)
Y top limit.

0	Not included in list of *BB* values
1	Is included in list of *BB* values.

RES2 (GBIT2)
Reserved value, must be 0.

GSSB Parameter - BB[LEN]

BB[LEN] (GROSOL)
Boundary values.

Set Segment Boundary - Parameters

LEN (GLENGTH1)
Length of following data.

RES (GBIT8)
Reserved value, must be 0.

MASK
Values:

RES1 (GBIT2)
Reserved value, must be 0.

XL (GBIT1)
X left limit.

0	Not included in list of *BB* values
1	Is included in list of *BB* values.

XR (GBIT1)
X right limit.

0	Not included in list of *BB* values
1	Is included in list of *BB* values.

YB (GBIT1)
Y bottom limit.

0	Not included in list of *BB* values
1	Is included in list of *BB* values.

YT (GBIT1)
Y top limit.

0	Not included in list of *BB* values
1	Is included in list of *BB* values.

RES2 (GBIT2)
Reserved value, must be 0.

BB[LEN] (GROSOL)
Boundary values.

Set Segment Boundary - Remarks

The order is only valid in a root-segment prolog.

Set Segment Boundary - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Stroke Line Width/Push and Set Stroke Line Width

Set Stroke Line Width/Push and Set Stroke Line Width - Syntax

These orders set the current stroke line-width attribute.

Set Stroke Line Width (GSSLW)
X'15' (LEN, FLAGS, RES, STRWIDTH)

Push and Set Stroke Line Width (GPSSLW)
X'55' (LEN, FLAGS, RES, STRWIDTH)

GSSLW/GPSSLW Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSSLW/GPSSLW Parameter - FLAGS

FLAGS

DEFLT (GBIT1)
Values:

0	Set to value
1	Set to drawing default.

RES (GBIT7)
Reserved value, must be 0.

GSSLW/GPSSLW Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GSSLW/GPSSLW Parameter - STRWIDTH

STRWIDTH (GROSOL)

Value for stroke width.

Set Stroke Line Width/Push and Set Stroke Line Width - Parameters

LEN (GLENGTH1)
Length of following data.

FLAGS

DEFLT (GBIT1)
Values:

0	Set to value
1	Set to drawing default.

RES (GBIT7)
Reserved value, must be 0.

RES (GBIT8)
Reserved value, must be 0.

STRWIDTH (GROSOL)
Value for stroke width.

Set Stroke Line Width/Push and Set Stroke Line Width - Topics

Select an item:
Syntax
Parameters
Glossary

Set Text Alignment/Push and Set Text Alignment

Set Text Alignment/Push and Set Text Alignment - Syntax

These orders set, or push and set, the value of the current text alignment attribute.

Set Text Alignment (GSCSA)
X'10' (LEN, HORIZ, VERT)

Push and Set Text Alignment (GPSCSA)
X'50' (LEN, HORIZ, VERT)

GSCSA/GPSCSA Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSCSA/GPSCSA Parameter - HORIZ

HORIZ (GUCHAR)
Horizontal alignment as follows:

0x00	Drawing default.	
0x01	Normal alignment. The alignment assumed depends on the current character direction:	
	Left to right	Left alignment.
	Top to bottom	Center alignment.
	Right to left	Right alignment.
	Bottom to top	Center alignment.
0x02	Left alignment. The string is aligned on the left edge of its leftmost character.	
0x03	Center alignment. The string is aligned on the arithmetic mean of left and right.	
0x04	Right alignment. The string is aligned on the right edge of its rightmost character.	
0xFF	Standard alignment. The alignment assumed depends on the current character direction:	
	Left to right	Left alignment.
	Top to bottom	Left alignment.
	Right to left	Right alignment.
	Bottom to top	Left alignment.

GSCSA/GPSCSA Parameter - VERT

VERT (GUCHAR)
Vertical alignment as follows:

0x00	Drawing default.	
0x01	Normal alignment. The alignment assumed depends on the current character direction:	
	Left to right	Base alignment.
	Top to bottom	Top alignment.
	Right to left	Base alignment.
	Bottom to top	Bottom alignment.
0x02	Top Alignment. The string is aligned on the top edge of its topmost character.	
0x03	Reserved.	

0x04	Half alignment. The string is aligned on the arithmetic mean of top and bottom.
0x05	Base alignment. The string is aligned on the base of its bottom character.
0x06	Bottom Alignment. The string is aligned on the bottom edge of its bottom character.
0xFF	Standard alignment. The alignment assumed depends on the current character direction:

Left to right	Bottom alignment.
Top to bottom	Top alignment.
Right to left	Bottom alignment.
Bottom to top	Bottom alignment.

Set Text Alignment/Push and Set Text Alignment - Parameters

LEN (GLENGTH1)
Length of following data.

HORIZ (GUCHAR)
Horizontal alignment as follows:

0x00	Drawing default.
0x01	Normal alignment. The alignment assumed depends on the current character direction:

Left to right	Left alignment.
Top to bottom	Center alignment.
Right to left	Right alignment.
Bottom to top	Center alignment.

0x02	Left alignment. The string is aligned on the left edge of its leftmost character.
0x03	Center alignment. The string is aligned on the arithmetic mean of left and right.
0x04	Right alignment. The string is aligned on the right edge of its rightmost character.
0xFF	Standard alignment. The alignment assumed depends on the current character direction:

Left to right	Left alignment.
Top to bottom	Left alignment.
Right to left	Right alignment.
Bottom to top	Left alignment.

VERT (GUCHAR)
Vertical alignment as follows:

0x00	Drawing default.
0x01	Normal alignment. The alignment assumed depends on the current character direction:

Left to right	Base alignment.
Top to bottom	Top alignment.
Right to left	Base alignment.
Bottom to top	Bottom alignment.

0x02	Top Alignment. The string is aligned on the top edge of its topmost character.
0x03	Reserved.
0x04	Half alignment. The string is aligned on the arithmetic mean of top and bottom.
0x05	Base alignment. The string is aligned on the base of its bottom character.

0x06	Bottom Alignment. The string is aligned on the bottom edge of its bottom character.	
0xFF	Standard alignment. The alignment assumed depends on the current character direction:	
	Left to right	Bottom alignment.
	Top to bottom	Top alignment.
	Right to left	Bottom alignment.
	Bottom to top	Bottom alignment.

Set Text Alignment/Push and Set Text Alignment - Remarks

The value of the current text alignment attribute is pushed on to the Segment Call stack by the Push and Set order only. Both orders set the value of the current text alignment attribute to the value specified in the order.

Set Text Alignment/Push and Set Text Alignment - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Set Viewing Transform

Set Viewing Transform - Syntax

This order sets the current viewing transform.

Set Viewing Transform (GSTV)
X'31' (LEN, RES, FLAGS, MASK, MX[LEN])

GSTV Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSTV Parameter - RES

RES (GBIT8)
Reserved value, must be 0.

GSTV Parameter - FLAGS

FLAGS
Values:

RES1 (GBIT5)
Reserved value, must be 0.

CONTROL (GBIT1)
Values:

0	Concatenate before drawing default
1	Concatenate before the current viewing transform.

RES2 (GBIT2)
Reserved value, must be 0.

GSTV Parameter - MASK

MASK (GBIT16)
Load mask.

GSTV Parameter - MX[LEN]

MX[LEN] (GROSOL)
Matrix values.

Set Viewing Transform - Parameters

LEN (GLENGTH1)
Length of following data.

RES (GBIT8)
Reserved value, must be 0.

FLAGS
Values:

RES1 (GBIT5)
Reserved value, must be 0.

CONTROL (GBIT1)
Values:

0	Concatenate before drawing default
1	Concatenate before the current viewing transform.

RES2 (GBIT2)
Reserved value, must be 0.

MASK (GBIT16)
Load mask.

MX[LEN] (GROSOL)
Matrix values.

Set Viewing Transform - Topics

Select an item:
Syntax
Parameters
Glossary

Set Viewing Window/Push and Set Viewing Window

Set Viewing Window/Push and Set Viewing Window - Syntax

These orders set, or push and set, the current viewing window.

Set Viewing Window (GSVW)
X'27' (LEN, FLAG, MASK, WW[LEN])

Push and Set Viewing Window (GPSVW)
X'67' (LEN, FLAG, MASK, WW[LEN])

GSVW/GPSVW Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSVW/GPSVW Parameter - FLAG

FLAG
Values:

REPLACE (GBIT1)
Values:

0	Intersect with current window
1	Replace current with new window.

RES (GBIT7)
Reserved value, must be 0.

GSVW/GPSVW Parameter - MASK

MASK
Values:

RES1 (GBIT2)
Reserved value, must be 0.

XL (GBIT1)
X left limit.

0	Not included in list of *WW* values
1	Is included in list of *WW* values

XR (GBIT1)
X right limit.

0	Not included in list of *WW* values
1	Is included in list of *WW* values

YB (GBIT1)
Y bottom limit.

0	Not included in list of *WW* values
1	Is included in list of *WW* values

YT (GBIT1)
Y top limit.

0	Not included in list of *WW* values
1	Is included in list of *WW* values

RES2 (GBIT2)
Reserved value, must be 0.

GSVW/GPSVW Parameter - WW[LEN]

WW[LEN] (GROSOL)
Window values.

Set Viewing Window/Push and Set Viewing Window - Parameters

LEN (GLENGTH1)
Length of following data.

FLAG
Values:

REPLACE (GBIT1)
Values:

0	Intersect with current window
1	Replace current with new window.

RES (GBIT7)
Reserved value, must be 0.

MASK
Values:

RES1 (GBIT2)
Reserved value, must be 0.

XL (GBIT1)
X left limit.

0	Not included in list of *WW* values
1	Is included in list of *WW* values

XR (GBIT1)
X right limit.

0	Not included in list of *WW* values
1	Is included in list of *WW* values

YB (GBIT1)
Y bottom limit.

0	Not included in list of *WW* values
1	Is included in list of *WW* values

YT (GBIT1)
Y top limit.

0	Not included in list of *WW* values
1	Is included in list of *WW* values

RES2 (GBIT2)
Reserved value, must be 0.

WW[LEN] (GROSOL)
Window values.

Set Viewing Window/Push and Set Viewing Window - Remarks

The value of the current viewing window is pushed on to the Segment Call stack by the Push and Set order only. Both orders then set the current viewing window using the values in the order.

Set Viewing Window/Push and Set Viewing Window - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Sharp Fillet at Given Position/Sharp Fillet at Current Position

Sharp Fillet at Given Position/Sharp Fillet at Current Position - Syntax

This order generates a curve that starts at a given position, and uses points P1 and P2, together with the sharpness specification S1.

Sharp Fillet at Given Position (GSFLT)
X'E4' (LEN, P0, P1, P2, P3, P4, PN-1, PN, S1, S2, SN/2)

Sharp Fillet at Current Position (GCSFLT)
X'A4' (LEN, P1, P2, P3, P4, PN-1, PN, S1, S2, SN/2)

GSFLT/GCSFLT Parameter - LEN

LEN (GLENGTH1)
Length of following data.

GSFLT/GCSFLT Parameter - P0

P0 (GPOINT)
Coordinate data of first curve start.

This parameter is only present in a Sharp Fillet at Given Position order.

GSFLT/GCSFLT Parameter - P1

P1 (GPOINT)
Coordinate data of first curve control point.

GSFLT/GCSFLT Parameter - P2

P2 (GPOINT)
Coordinate data of first curve end.

GSFLT/GCSFLT Parameter - P3

P3 (GPOINT)
Coordinate data of second curve control point.

GSFLT/GCSFLT Parameter - P4

P4 (GPOINT)
Coordinate data of second curve end.

GSFLT/GCSFLT Parameter - PN-1

PN-1 (GPOINT)
Coordinate data of last curve control point.

GSFLT/GCSFLT Parameter - PN

PN (GPOINT)
Coordinate data of last curve end.

GSFLT/GCSFLT Parameter - S1

S1 (GROF)
Sharpness specification of first curve.

GSFLT/GCSFLT Parameter - S2

S2 (GROF)
Sharpness specification of second curve.

GSFLT/GCSFLT Parameter - SN/2

SN/2 (GROF)
Sharpness specification of last curve.

Sharp Fillet at Given Position/Sharp Fillet at Current Position - Parameters

LEN (GLENGTH1)
Length of following data.

P0 (GPOINT)
Coordinate data of first curve start.

This parameter is only present in a Sharp Fillet at Given Position order.

P1 (GPOINT)
Coordinate data of first curve control point.

P2 (GPOINT)
Coordinate data of first curve end.

P3 (GPOINT)
Coordinate data of second curve control point.

P4 (GPOINT)
Coordinate data of second curve end.

PN-1 (GPOINT)
Coordinate data of last curve control point.

PN (GPOINT)
Coordinate data of last curve end.

S1 (GROF)
Sharpness specification of first curve.

S2 (GROF)
Sharpness specification of second curve.

SN/2 (GROF)
Sharpness specification of last curve.

Sharp Fillet at Given Position/Sharp Fillet at Current Position - Remarks

Further points are used in groups of two to form a polycurve.

Sharp Fillet at Given Position/Sharp Fillet at Current Position - Topics

Select an item:
Syntax
Parameters
Remarks
Glossary

Graphics Orders Data Types

All data types are in Intel** format, unless noted otherwise.

GBIT1

1-bit field.

GBIT2

2-bit field.

GBIT4

4-bit field.

GBIT5

5-bit field.

GBIT6

6-bit field.

GBIT7

7-bit field.

GBIT8

8-bit field.

GBIT16

16-bit field.

GBIT32

32-bit field.

GCHAR

Signed 1-byte integer value.

GDELPOINT

Offset point structure.

```
  GCHAR     dx;  /*  x coordinate offset. */
```

```
GCHAR     dy;  /*  y coordinate offset. */
```

GDELPOINT Field - dx

dx (GCHAR)
x coordinate offset.

GDELPOINT Field - dy

dy (GCHAR)
y coordinate offset.

GFIXED

Signed integer fraction (16:16). (This can be treated as a GLONG where the value has been multiplied by 65536.)

GFIXEDS

Signed integer fraction (8:8), which can be treated as a GSHORT data type, where the value has been multiplied by 256.

```
GCHAR      Integer;   /*  Integral component. */
GUCHAR     Fraction;  /*  Fractional component. */
```

GFIXEDS Field - Integer

Integer (GCHAR)
Integral component.

GFIXEDS Field - Fraction

Fraction (GUCHAR)
Fractional component.

GHBITMAP

Bit-map handle, which is the same as GULONG.

GINDATT

Individual attribute value. For the attribute types color and background color, this is the same as GINDEX3. For the attribute types mix and background color, this is the same as GUCHAR.

GINDEX3

Unsigned 3-byte integer value.

GLENGTH1

1-byte length.

GLENGTH2

2-byte length, in S/370 format; that is, the high-order byte precedes the low-order byte in storage.

GLONG

Signed 4-byte integer value.

GPOINT

Point structure.

```
GROSOL     x;  /*  x coordinate. */
GROSOL     y;  /*  y coordinate. */
```

GPOINT Field - x

x (GROSOL)
x coordinate.

GPOINT Field - y

y (GROSOL)
y coordinate.

GPOINTB

Point in bit-map structure.

```
GLONG      x;  /*  x coordinate. */
GLONG      y;  /*  y coordinate. */
```

GPOINTB Field - x

x (GLONG)
x coordinate.

GPOINTB Field - y

y (GLONG)
y coordinate.

GPOLYS

Array of Polygons. Each element of the array is a 16 bit count of the number of vertices, followed by the vertex coordinates.

GREAL

Real (single precision floating point).

This data type is in Intel format.

GROF

Number representation which is the same as the GFIXED data type.

GROFUFS

Number representation which is either GFIXED, GUFIXEDS or GREAL data type, depending on the presentation-space format.

GROUFS

Number representation which is either the GUFIXEDS or GREAL data type, depending on the presentation-space format.

GROL

Number representation, which is the same as the GLONG data type.

GROSOL

Number representation which is either the GSHORT or the GLONG data type, depending on the presentation-space format; see PS_FORMAT in the *flOptions* parameter of the GpiCreatePS function.

GROUL

Number representation, which is the same as the GULONG data type.

GSHORT

Signed 2-byte integer value.

GSHORT370

Signed 2-byte integer value, in S/370 format (that is, the high-order byte precedes the low-order byte in storage).

GSTR

String with an explicit length count.

GUCHAR

Unsigned 1-byte integer value.

GUFIXEDS

Unsigned integer fraction (8:8) which can be treated as a GUSHORT data type, where the value has been multiplied by 256.

GULONG

Unsigned 4-byte integer value.

GULONG370

Unsigned 4-byte integer value, in S/370 format (that is, the high-order byte first, the low-order byte last in storage).

GUNDF

Undefined string of 8-bit bytes.

GUNDF1

Undefined 8-bit byte.

GUSHORT

Unsigned 2-byte integer value.

GUSHORT370

Unsigned 2-byte integer value, in S/370 format; that is, the high-order byte precedes the low-order byte in storage.

Errors

Error codes are furnished numerically and alphabetically.

For a listing of error codes by number, see Error Number and Name.

For a listing of error codes and their explanations, see Error Name and Explanation.

Error Number and Name

This section lists PM errors returned by WinGetLastError in order of their error numbers. For explanations of these errors, see Error Name and Explanation.

Error Number	Error Constant
0x0000	PMERR_OK
0x0000	PMERR_OK
0x0000	PMERR_OK
0x0836	NERR_NetNotStarted
0x0845	NERR_RedirectedPath
0x084B	NERR_BufTooSmall

0x085E	NERR_InvalidAPI
0x0866	NERR_QNotFound
0x0867	NERR_JobNotFound
0x0868	NERR_DestNotFound
0x0869	NERR_DestExists
0x086A	NERR_QExists
0x086B	NERR_QNoRoom
0x086C	NERR_JobNoRoom
0x086D	NERR_DestNoRoom
0x086E	NERR_DestIdle
0x086F	NERR_DestInvalidOp
0x0871	NERR_SpoolerNotLoaded
0x0872	NERR_DestInvalidState
0x0874	NERR_JobInvalidState
0x0875	NERR_SpoolNoMemory
0x0876	NERR_DriverNotFound
0x0877	NERR_DataTypeInvalid
0x0878	NERR_ProcNotFound
0x0925	NERR_BadDev
0x0927	NERR_CommDevInUse
0x092F	NERR_InvalidComputer
0x0961	NERR_OpenFiles
0x0965	NERR_LocalDrive
0x1001	PMERR_INVALID_HWND
0x1001	HMERR_NO_FRAME_WND_IN_CHAIN
0x1002	PMERR_INVALID_HMQ
0x1002	HMERR_INVALID_ASSOC_APP_WND
0x1003	PMERR_PARAMETER_OUT_OF_RANGE
0x1003	HMERR_INVALID_ASSOC_HELP_INST
0x1004	PMERR_WINDOW_LOCK_UNDERFLOW
0x1004	HMERR_INVALID_DESTROY_HELP_INST
0x1005	PMERR_WINDOW_LOCK_OVERFLOW
0x1005	HMERR_NO_HELP_INST_IN_CHAIN
0x1006	PMERR_BAD_WINDOW_LOCK_COUNT
0x1006	HMERR_INVALID_HELP_INSTANCE_HDL
0x1007	PMERR_WINDOW_NOT_LOCKED
0x1007	HMERR_INVALID_QUERY_APP_WND
0x1008	PMERR_INVALID_SELECTOR
0x1008	HMERR_HELP_INST_CALLED_INVALID
0x1009	PMERR_CALL_FROM_WRONG_THREAD
0x1009	HMERR_HELPTABLE_UNDEFINE
0x100A	PMERR_RESOURCE_NOT_FOUND
0x100A	HMERR_HELP_INSTANCE_UNDEFINE
0x100B	PMERR_INVALID_STRING_PARM
0x100B	HMERR_HELPITEM_NOT_FOUND
0x100C	PMERR_INVALID_HHEAP
0x100C	HMERR_INVALID_HELPSUBITEM_SIZE
0x100D	PMERR_INVALID_HEAP_POINTER
0x100D	HMERR_HELPSUBITEM_NOT_FOUND
0x100E	PMERR_INVALID_HEAP_SIZE_PARM
0x100F	PMERR_INVALID_HEAP_SIZE
0x1010	PMERR_INVALID_HEAP_SIZE_WORD
0x1011	PMERR_HEAP_OUT_OF_MEMORY
0x1012	PMERR_HEAP_MAX_SIZE_REACHED
0x1013	PMERR_INVALID_HATOMTBL
0x1014	PMERR_INVALID_ATOM
0x1015	PMERR_INVALID_ATOM_NAME
0x1016	PMERR_INVALID_INTEGER_ATOM
0x1017	PMERR_ATOM_NAME_NOT_FOUND
0x1018	PMERR_QUEUE_TOO_LARGE
0x1019	PMERR_INVALID_FLAG
0x101A	PMERR_INVALID_HACCEL
0x101B	PMERR_INVALID_HPTR
0x101C	PMERR_INVALID_HENUM
0x101D	PMERR_INVALID_SRC_CODEPAGE
0x101E	PMERR_INVALID_DST_CODEPAGE
0x101F	PMERR_UNKNOWN_COMPONENT_ID
0x1020	PMERR_UNKNOWN_ERROR_CODE
0x1021	PMERR_SEVERITY_LEVELS
0x1034	PMERR_INVALID_RESOURCE_FORMAT
0x1035	WINDBG_WINDOW_UNLOCK_WAIT
0x1036	PMERR_NO_MSG_QUEUE
0x1037	PMERR_CANNOT_SET_FOCUS

0x1038	PMERR_QUEUE_FULL
0x1039	PMERR_LIBRARY_LOAD_FAILED
0x103A	PMERR_PROCEDURE_LOAD_FAILED
0x103B	PMERR_LIBRARY_DELETE_FAILED
0x103C	PMERR_PROCEDURE_DELETE_FAILED
0x103D	PMERR_ARRAY_TOO_LARGE
0x103E	PMERR_ARRAY_TOO_SMALL
0x103F	PMERR_DATATYPE_ENTRY_BAD_INDEX
0x1040	PMERR_DATATYPE_ENTRY_CTL_BAD
0x1041	PMERR_DATATYPE_ENTRY_CTL_MISS
0x1042	PMERR_DATATYPE_ENTRY_INVALID
0x1043	PMERR_DATATYPE_ENTRY_NOT_NUM
0x1044	PMERR_DATATYPE_ENTRY_NOT_OFF
0x1045	PMERR_DATATYPE_INVALID
0x1046	PMERR_DATATYPE_NOT_UNIQUE
0x1047	PMERR_DATATYPE_TOO_LONG
0x1048	PMERR_DATATYPE_TOO_SMALL
0x1049	PMERR_DIRECTION_INVALID
0x104A	PMERR_INVALID_HAB
0x104D	PMERR_INVALID_HSTRUCT
0x104E	PMERR_LENGTH_TOO_SMALL
0x104F	PMERR_MSGID_TOO_SMALL
0x1050	PMERR_NO_HANDLE_ALLOC
0x1051	PMERR_NOT_IN_A_PM_SESSION
0x1052	PMERR_MSG_QUEUE_ALREADY_EXISTS
0x1055	PMERR_OLD_RESOURCE
0x1056	PMERR_WPDSERVER_IS_ACTIVE
0x1057	PMERR_WPDSERVER_NOT_STARTED
0x1058	PMERR_SOMDD_IS_ACTIVE
0x1059	PMERR_SOMDD_NOT_STARTED
0x1101	PMERR_INVALID_PIB
0x1102	PMERR_INSUFF_SPACE_TO_ADD
0x1103	PMERR_INVALID_GROUP_HANDLE
0x1104	PMERR_DUPLICATE_TITLE
0x1105	PMERR_INVALID_TITLE
0x1106	PMERR_INVALID_TARGET_HANDLE
0x1107	PMERR_HANDLE_NOT_IN_GROUP
0x1108	PMERR_INVALID_PATH_STATEMENT
0x1109	PMERR_NO_PROGRAM_FOUND
0x110A	PMERR_INVALID_BUFFER_SIZE
0x110B	PMERR_BUFFER_TOO_SMALL
0x110C	PMERR_PL_INITIALISATION_FAIL
0x110D	PMERR_CANT_DESTROY_SYS_GROUP
0x110E	PMERR_INVALID_TYPE_CHANGE
0x110F	PMERR_INVALID_PROGRAM_HANDLE
0x1110	PMERR_NOT_CURRENT_PL_VERSION
0x1111	PMERR_INVALID_CIRCULAR_REF
0x1112	PMERR_MEMORY_ALLOCATION_ERR
0x1113	PMERR_MEMORY_DEALLOCATION_ERR
0x1114	PMERR_TASK_HEADER_TOO_BIG
0x1115	PMERR_INVALID_INI_FILE_HANDLE
0x1116	PMERR_MEMORY_SHARE
0x1117	PMERR_OPEN_QUEUE
0x1118	PMERR_CREATE_QUEUE
0x1119	PMERR_WRITE_QUEUE
0x111A	PMERR_READ_QUEUE
0x111B	PMERR_CALL_NOT_EXECUTED
0x111C	PMERR_UNKNOWN_APIPKT
0x111D	PMERR_INITHREAD_EXISTS
0x111E	PMERR_CREATE_THREAD
0x111F	PMERR_NO_HK_PROFILE_INSTALLED
0x1120	PMERR_INVALID_DIRECTORY
0x1121	PMERR_WILDCARD_IN_FILENAME
0x1122	PMERR_FILENAME_BUFFER_FULL
0x1123	PMERR_FILENAME_TOO_LONG
0x1124	PMERR_INI_FILE_IS_SYS_OR_USER
0x1125	PMERR_BROADCAST_PLMSG
0x1126	PMERR_190_INIT_DONE
0x1127	PMERR_HMOD_FOR_PMSHAPI
0x1128	PMERR_SET_HK_PROFILE
0x1129	PMERR_API_NOT_ALLOWED
0x112A	PMERR_INI_STILL_OPEN
0x112B	PMERR_PROGDETAILS_NOT_IN_INI

0x112C	PMERR_PIBSTRUCT_NOT_IN_INI
0x112D	PMERR_INVALID_DISKPROGDETAILS
0x112E	PMERR_PROGDETAILS_READ_FAILURE
0x112F	PMERR_PROGDETAILS_WRITE_FAILURE
0x1130	PMERR_PROGDETAILS_QSIZE_FAILURE
0x1131	PMERR_INVALID_PROGDETAILS
0x1132	PMERR_SHEPROFILEHOOK_NOT_FOUND
0x1133	PMERR_190PLCONVERTED
0x1134	PMERR_FAILED_TO_CONVERT_INI_PL
0x1135	PMERR_PMSHAPI_NOT_INITIALISED
0x1136	PMERR_INVALID_SHELL_API_HOOK_ID
0x1200	PMERR_DOS_ERROR
0x1201	PMERR_NO_SPACE
0x1202	PMERR_INVALID_SWITCH_HANDLE
0x1203	PMERR_NO_HANDLE
0x1204	PMERR_INVALID_PROCESS_ID
0x1205	PMERR_NOT_SHELL
0x1206	PMERR_INVALID_WINDOW
0x1207	PMERR_INVALID_POST_MSG
0x1208	PMERR_INVALID_PARAMETERS
0x1208	PMERR_INVALID_PARAMETERS
0x1209	PMERR_INVALID_PROGRAM_TYPE
0x120A	PMERR_NOT_EXTENDED_FOCUS
0x120B	PMERR_INVALID_SESSION_ID
0x120C	PMERR_SMG_INVALID_ICON_FILE
0x120D	PMERR_SMG_ICON_NOT_CREATED
0x120E	PMERR_SHL_DEBUG
0x1301	PMERR_OPENING_INI_FILE
0x1302	PMERR_INI_FILE_CORRUPT
0x1303	PMERR_INVALID_PARM
0x1304	PMERR_NOT_IN_IDX
0x1305	PMERR_NO_ENTRIES_IN_GROUP
0x1306	PMERR_INI_WRITE_FAIL
0x1307	PMERR_IDX_FULL
0x1308	PMERR_INI_PROTECTED
0x1309	PMERR_MEMORY_ALLOC
0x130A	PMERR_INI_INIT_ALREADY_DONE
0x130B	PMERR_INVALID_INTEGER
0x130C	PMERR_INVALID_ASCIIZ
0x130D	PMERR_CAN_NOT_CALL_SPOOLER
0x130D	PMERR_VALIDATION_REJECTED
0x1401	PMERR_WARNING_WINDOW_NOT_KILLED
0x1402	PMERR_ERROR_INVALID_WINDOW
0x1403	PMERR_ALREADY_INITIALIZED
0x1405	PMERR_MSG_PROG_NO_MOU
0x1406	PMERR_MSG_PROG_NON_RECOV
0x1407	PMERR_WINCONV_INVALID_PATH
0x1408	PMERR_PI_NOT_INITIALISED
0x1409	PMERR_PL_NOT_INITIALISED
0x140A	PMERR_NO_TASK_MANAGER
0x140B	PMERR_SAVE_NOT_IN_PROGRESS
0x140C	PMERR_NO_STACK_SPACE
0x140D	PMERR_INVALID_COLR_FIELD
0x140E	PMERR_INVALID_COLR_VALUE
0x140F	PMERR_COLR_WRITE
0x1501	PMERR_TARGET_FILE_EXISTS
0x1502	PMERR_SOURCE_SAME_AS_TARGET
0x1503	PMERR_SOURCE_FILE_NOT_FOUND
0x1504	PMERR_INVALID_NEW_PATH
0x1505	PMERR_TARGET_FILE_NOT_FOUND
0x1506	PMERR_INVALID_DRIVE_NUMBER
0x1507	PMERR_NAME_TOO_LONG
0x1508	PMERR_NOT_ENOUGH_ROOM_ON_DISK
0x1509	PMERR_NOT_ENOUGH_MEM
0x150B	PMERR_LOG_DRV_DOES_NOT_EXIST
0x150C	PMERR_INVALID_DRIVE
0x150D	PMERR_ACCESS_DENIED
0x150E	PMERR_NO_FIRST_SLASH
0x150F	PMERR_READ_ONLY_FILE
0x151F	PMERR_GROUP_PROTECTED
0x152F	PMERR_INVALID_PROGRAM_CATEGORY
0x1530	PMERR_INVALID_APPL
0x1531	PMERR_CANNOT_START

0x1532	PMERR_STARTED_IN_BACKGROUND
0x1533	PMERR_INVALID_HAPP
0x1534	PMERR_CANNOT_STOP
0x1601	PMERR_INTERNAL_ERROR_1
0x1602	PMERR_INTERNAL_ERROR_2
0x1603	PMERR_INTERNAL_ERROR_3
0x1604	PMERR_INTERNAL_ERROR_4
0x1605	PMERR_INTERNAL_ERROR_5
0x1606	PMERR_INTERNAL_ERROR_6
0x1607	PMERR_INTERNAL_ERROR_7
0x1608	PMERR_INTERNAL_ERROR_8
0x1609	PMERR_INTERNAL_ERROR_9
0x160A	PMERR_INTERNAL_ERROR_10
0x160B	PMERR_INTERNAL_ERROR_11
0x160C	PMERR_INTERNAL_ERROR_12
0x160D	PMERR_INTERNAL_ERROR_13
0x160E	PMERR_INTERNAL_ERROR_14
0x160F	PMERR_INTERNAL_ERROR_15
0x1610	PMERR_INTERNAL_ERROR_16
0x1611	PMERR_INTERNAL_ERROR_17
0x1612	PMERR_INTERNAL_ERROR_18
0x1613	PMERR_INTERNAL_ERROR_19
0x1614	PMERR_INTERNAL_ERROR_20
0x1615	PMERR_INTERNAL_ERROR_21
0x1616	PMERR_INTERNAL_ERROR_22
0x1617	PMERR_INTERNAL_ERROR_23
0x1618	PMERR_INTERNAL_ERROR_24
0x1619	PMERR_INTERNAL_ERROR_25
0x161A	PMERR_INTERNAL_ERROR_26
0x161B	PMERR_INTERNAL_ERROR_27
0x161C	PMERR_INTERNAL_ERROR_28
0x161D	PMERR_INTERNAL_ERROR_29
0x1630	PMERR_INVALID_FREE_MESSAGE_ID
0x1641	PMERR_FUNCTION_NOT_SUPPORTED
0x1642	PMERR_INVALID_ARRAY_COUNT
0x1643	PMERR_INVALID_LENGTH
0x1644	PMERR_INVALID_BUNDLE_TYPE
0x1645	PMERR_INVALID_PARAMETER
0x1646	PMERR_INVALID_NUMBER_OF_PARMS
0x1647	PMERR_GREATER_THAN_64K
0x1648	PMERR_INVALID_PARAMETER_TYPE
0x1649	PMERR_NEGATIVE_STRCOND_DIM
0x164A	PMERR_INVALID_NUMBER_OF_TYPES
0x164B	PMERR_INCORRECT_HSTRUCT
0x164C	PMERR_INVALID_ARRAY_SIZE
0x164D	PMERR_INVALID_CONTROL_DATATYPE
0x164E	PMERR_INCOMPLETE_CONTROL_SEQU
0x164F	PMERR_INVALID_DATATYPE
0x1650	PMERR_INCORRECT_DATATYPE
0x1651	PMERR_NOT_SELF_DESCRIBING_DTYP
0x1652	PMERR_INVALID_CTRL_SEQ_INDEX
0x1653	PMERR_INVALID_TYPE_FOR_LENGTH
0x1654	PMERR_INVALID_TYPE_FOR_OFFSET
0x1655	PMERR_INVALID_TYPE_FOR_MPARAM
0x1656	PMERR_INVALID_MESSAGE_ID
0x1657	PMERR_C_LENGTH_TOO_SMALL
0x1658	PMERR_APPL_STRUCTURE_TOO_SMALL
0x1659	PMERR_INVALID_ERRORINFO_HANDLE
0x165A	PMERR_INVALID_CHARACTER_INDEX
0x1700	WPERR_PROTECTED_CLASS
0x1701	WPERR_INVALID_CLASS
0x1702	WPERR_INVALID_SUPERCLASS
0x1703	WPERR_NO_MEMORY
0x1704	WPERR_SEMAPHORE_ERROR
0x1705	WPERR_BUFFER_TOO_SMALL
0x1706	WPERR_CLSLOADMOD_FAILED
0x1707	WPERR_CLSPROCADDR_FAILED
0x1708	WPERR_OBJWORD_LOCATION
0x1709	WPERR_INVALID_OBJECT
0x170A	WPERR_MEMORY_CLEANUP
0x170B	WPERR_INVALID_MODULE
0x170C	WPERR_INVALID_OLDCLASS
0x170D	WPERR_INVALID_NEWCLASS

0x170E	WPERR_NOT_IMMEDIATE_CHILD
0x170F	WPERR_NOT_WORKPLACE_CLASS
0x1710	WPERR_CANT_REPLACE_METACLS
0x1711	WPERR_INI_FILE_WRITE
0x1712	WPERR_INVALID_FOLDER
0x1713	WPERR_BUFFER_OVERFLOW
0x1714	WPERR_OBJECT_NOT_FOUND
0x1715	WPERR_INVALID_HFIND
0x1716	WPERR_INVALID_COUNT
0x1717	WPERR_INVALID_BUFFER
0x1718	WPERR_ALREADY_EXISTS
0x1719	WPERR_INVALID_FLAGS
0x1720	WPERR_INVALID_OBJECTID
0x1721	WPERR_INVALID_TARGET_OBJECT
0x1F00	PMERR_NOT_DRAGGING
0x2001	PMERR_ALREADY_IN_AREA
0x2001	HMERR_INDEX_NOT_FOUND
0x2002	PMERR_ALREADY_IN_ELEMENT
0x2002	HMERR_CONTENT_NOT_FOUND
0x2003	PMERR_ALREADY_IN_PATH
0x2003	HMERR_OPEN_LIB_FILE
0x2004	PMERR_ALREADY_IN_SEG
0x2004	HMERR_READ_LIB_FILE
0x2005	PMERR_AREA_INCOMPLETE
0x2005	HMERR_CLOSE_LIB_FILE
0x2006	PMERR_BASE_ERROR
0x2006	HMERR_INVALID_LIB_FILE
0x2007	PMERR_BITBLT_LENGTH_EXCEEDED
0x2007	HMERR_NO_MEMORY
0x2008	PMERR_BITMAP_IN_USE
0x2008	HMERR_ALLOCATE_SEGMENT
0x2009	PMERR_BITMAP_IS_SELECTED
0x2009	HMERR_FREE_MEMORY
0x200A	PMERR_BITMAP_NOT_FOUND
0x200B	PMERR_BITMAP_NOT_SELECTED
0x200C	PMERR_BOUNDS_OVERFLOW
0x200D	PMERR_CALLED_SEG_IS_CHAINED
0x200E	PMERR_CALLED_SEG_IS_CURRENT
0x200F	PMERR_CALLED_SEG_NOT_FOUND
0x2010	PMERR_CANNOT_DELETE_ALL_DATA
0x2010	HMERR_PANEL_NOT_FOUND
0x2011	PMERR_CANNOT_REPLACE_ELEMENT_0
0x2011	HMERR_DATABASE_NOT_OPEN
0x2012	PMERR_COL_TABLE_NOT_REALIZABLE
0x2013	PMERR_COL_TABLE_NOT_REALIZED
0x2013	HMERR_LOAD_DLL
0x2014	PMERR_COORDINATE_OVERFLOW
0x2015	PMERR_CORR_FORMAT_MISMATCH
0x2016	PMERR_DATA_TOO_LONG
0x2017	PMERR_DC_IS_ASSOCIATED
0x2018	PMERR_DESC_STRING_TRUNCATED
0x2019	PMERR_DEVICE_DRIVER_ERROR_1
0x201A	PMERR_DEVICE_DRIVER_ERROR_2
0x201B	PMERR_DEVICE_DRIVER_ERROR_3
0x201C	PMERR_DEVICE_DRIVER_ERROR_4
0x201D	PMERR_DEVICE_DRIVER_ERROR_5
0x201E	PMERR_DEVICE_DRIVER_ERROR_6
0x201F	PMERR_DEVICE_DRIVER_ERROR_7
0x2020	PMERR_DEVICE_DRIVER_ERROR_8
0x2021	PMERR_DEVICE_DRIVER_ERROR_9
0x2022	PMERR_DEVICE_DRIVER_ERROR_10
0x2023	PMERR_DEV_FUNC_NOT_INSTALLED
0x2024	PMERR_DOSOPEN_FAILURE
0x2025	PMERR_DOSREAD_FAILURE
0x2026	PMERR_DRIVER_NOT_FOUND
0x2027	PMERR_DUP_SEG
0x2028	PMERR_DYNAMIC_SEG_SEQ_ERROR
0x2029	PMERR_DYNAMIC_SEG_ZERO_INV
0x202A	PMERR_ELEMENT_INCOMPLETE
0x202B	PMERR_ESC_CODE_NOT_SUPPORTED
0x202C	PMERR_EXCEEDS_MAX_SEG_LENGTH
0x202D	PMERR_FONT_AND_MODE_MISMATCH
0x202E	PMERR_FONT_FILE_NOT_LOADED

0x202F	PMERR_FONT_NOT_LOADED
0x2030	PMERR_FONT_TOO_BIG
0x2031	PMERR_HARDWARE_INIT_FAILURE
0x2032	PMERR_HBITMAP_BUSY
0x2033	PMERR_HDC_BUSY
0x2034	PMERR_HRGN_BUSY
0x2035	PMERR_HUGE_FONTS_NOT_SUPPORTED
0x2036	PMERR_ID_HAS_NO_BITMAP
0x2037	PMERR_IMAGE_INCOMPLETE
0x2038	PMERR_INCOMPAT_COLOR_FORMAT
0x2039	PMERR_INCOMPAT_COLOR_OPTIONS
0x203A	PMERR_INCOMPATIBLE_BITMAP
0x203B	PMERR_INCOMPATIBLE_METAFILE
0x203C	PMERR_INCORRECT_DC_TYPE
0x203D	PMERR_INSUFFICIENT_DISK_SPACE
0x203E	PMERR_INSUFFICIENT_MEMORY
0x203F	PMERR_INV_ANGLE_PARM
0x2040	PMERR_INV_ARC_CONTROL
0x2041	PMERR_INV_AREA_CONTROL
0x2042	PMERR_INV_ARC_POINTS
0x2043	PMERR_INV_ATTR_MODE
0x2044	PMERR_INV_BACKGROUND_COL_ATTR
0x2045	PMERR_INV_BACKGROUND_MIX_ATTR
0x2046	PMERR_INV_BITBLT_MIX
0x2047	PMERR_INV_BITBLT_STYLE
0x2048	PMERR_INV_BITMAP_DIMENSION
0x2049	PMERR_INV_BOX_CONTROL
0x204A	PMERR_INV_BOX_ROUNDING_PARM
0x204B	PMERR_INV_CHAR_ANGLE_ATTR
0x204C	PMERR_INV_CHAR_DIRECTION_ATTR
0x204D	PMERR_INV_CHAR_MODE_ATTR
0x204E	PMERR_INV_CHAR_POS_OPTIONS
0x204F	PMERR_INV_CHAR_SET_ATTR
0x2050	PMERR_INV_CHAR_SHEAR_ATTR
0x2051	PMERR_INV_CLIP_PATH_OPTIONS
0x2052	PMERR_INV_CODEPAGE
0x2053	PMERR_INV_COLOR_ATTR
0x2054	PMERR_INV_COLOR_DATA
0x2055	PMERR_INV_COLOR_FORMAT
0x2056	PMERR_INV_COLOR_INDEX
0x2057	PMERR_INV_COLOR_OPTIONS
0x2058	PMERR_INV_COLOR_START_INDEX
0x2059	PMERR_INV_COORD_OFFSET
0x205A	PMERR_INV_COORD_SPACE
0x205B	PMERR_INV_COORDINATE
0x205C	PMERR_INV_CORRELATE_DEPTH
0x205D	PMERR_INV_CORRELATE_TYPE
0x205E	PMERR_INV_CURSOR_BITMAP
0x205F	PMERR_INV_DC_DATA
0x2060	PMERR_INV_DC_TYPE
0x2061	PMERR_INV_DEVICE_NAME
0x2062	PMERR_INV_DEV_MODES_OPTIONS
0x2063	PMERR_INV_DRAW_CONTROL
0x2064	PMERR_INV_DRAW_VALUE
0x2065	PMERR_INV_DRAWING_MODE
0x2066	PMERR_INV_DRIVER_DATA
0x2067	PMERR_INV_DRIVER_NAME
0x2068	PMERR_INV_DRAW_BORDER_OPTION
0x2069	PMERR_INV_EDIT_MODE
0x206A	PMERR_INV_ELEMENT_OFFSET
0x206B	PMERR_INV_ELEMENT_POINTER
0x206C	PMERR_INV_END_PATH_OPTIONS
0x206D	PMERR_INV_ESC_CODE
0x206E	PMERR_INV_ESCAPE_DATA
0x206F	PMERR_INV_EXTENDED_LCID
0x2070	PMERR_INV_FILL_PATH_OPTIONS
0x2071	PMERR_INV_FIRST_CHAR
0x2072	PMERR_INV_FONT_ATTRS
0x2073	PMERR_INV_FONT_FILE_DATA
0x2074	PMERR_INV_FOR_THIS_DC_TYPE
0x2075	PMERR_INV_FORMAT_CONTROL
0x2076	PMERR_INV_FORMS_CODE
0x2077	PMERR_INV_FONTDEF

0x2078	PMERR_INV_GEOM_LINE_WIDTH_ATTR
0x2079	PMERR_INV_GETDATA_CONTROL
0x207A	PMERR_INV_GRAPHICS_FIELD
0x207B	PMERR_INV_HBITMAP
0x207C	PMERR_INV_HDC
0x207D	PMERR_INV_HJOURNAL
0x207E	PMERR_INV_HMF
0x207F	PMERR_INV_HPS
0x2080	PMERR_INV_HRGN
0x2081	PMERR_INV_ID
0x2082	PMERR_INV_IMAGE_DATA_LENGTH
0x2083	PMERR_INV_IMAGE_DIMENSION
0x2084	PMERR_INV_IMAGE_FORMAT
0x2085	PMERR_INV_IN_AREA
0x2086	PMERR_INV_IN_CALLED_SEG
0x2087	PMERR_INV_IN_CURRENT_EDIT_MODE
0x2088	PMERR_INV_IN_DRAW_MODE
0x2089	PMERR_INV_IN_ELEMENT
0x208A	PMERR_INV_IN_IMAGE
0x208B	PMERR_INV_IN_PATH
0x208C	PMERR_INV_IN_RETAIN_MODE
0x208D	PMERR_INV_IN_SEG
0x208E	PMERR_INV_IN_VECTOR_SYMBOL
0x208F	PMERR_INV_INFO_TABLE
0x2090	PMERR_INV_JOURNAL_OPTION
0x2091	PMERR_INV_KERNING_FLAGS
0x2092	PMERR_INV_LENGTH_OR_COUNT
0x2093	PMERR_INV_LINE_END_ATTR
0x2094	PMERR_INV_LINE_JOIN_ATTR
0x2095	PMERR_INV_LINE_TYPE_ATTR
0x2096	PMERR_INV_LINE_WIDTH_ATTR
0x2097	PMERR_INV_LOGICAL_ADDRESS
0x2098	PMERR_INV_MARKER_BOX_ATTR
0x2099	PMERR_INV_MARKER_SET_ATTR
0x209A	PMERR_INV_MARKER_SYMBOL_ATTR
0x209B	PMERR_INV_MATRIX_ELEMENT
0x209C	PMERR_INV_MAX_HITS
0x209D	PMERR_INV_METAFILE
0x209E	PMERR_INV_METAFILE_LENGTH
0x209F	PMERR_INV_METAFILE_OFFSET
0x20A0	PMERR_INV_MICROPS_DRAW_CONTROL
0x20A1	PMERR_INV_MICROPS_FUNCTION
0x20A2	PMERR_INV_MICROPS_ORDER
0x20A3	PMERR_INV_MIX_ATTR
0x20A4	PMERR_INV_MODE_FOR_OPEN_DYN
0x20A5	PMERR_INV_MODE_FOR_REOPEN_SEG
0x20A6	PMERR_INV_MODIFY_PATH_MODE
0x20A7	PMERR_INV_MULTIPLIER
0x20A8	PMERR_INV_NESTED_FIGURES
0x20A9	PMERR_INV_OR_INCOMPAT_OPTIONS
0x20AA	PMERR_INV_ORDER_LENGTH
0x20AB	PMERR_INV_ORDERING_PARM
0x20AC	PMERR_INV_OUTSIDE_DRAW_MODE
0x20AD	PMERR_INV_PAGE_VIEWPORT
0x20AE	PMERR_INV_PATH_ID
0x20AF	PMERR_INV_PATH_MODE
0x20B0	PMERR_INV_PATTERN_ATTR
0x20B1	PMERR_INV_PATTERN_REF_PT_ATTR
0x20B2	PMERR_INV_PATTERN_SET_ATTR
0x20B3	PMERR_INV_PATTERN_SET_FONT
0x20B4	PMERR_INV_PICK_APERTURE_OPTION
0x20B5	PMERR_INV_PICK_APERTURE_POSN
0x20B6	PMERR_INV_PICK_APERTURE_SIZE
0x20B7	PMERR_INV_PICK_NUMBER
0x20B8	PMERR_INV_PLAY_METAFILE_OPTION
0x20B9	PMERR_INV_PRIMITIVE_TYPE
0x20BA	PMERR_INV_PS_SIZE
0x20BB	PMERR_INV_PUTDATA_FORMAT
0x20BC	PMERR_INV_QUERY_ELEMENT_NO
0x20BD	PMERR_INV_RECT
0x20BE	PMERR_INV_REGION_CONTROL
0x20BF	PMERR_INV_REGION_MIX_MODE
0x20C0	PMERR_INV_REPLACE_MODE_FUNC

0x20C1	PMERR_INV_RESERVED_FIELD
0x20C2	PMERR_INV_RESET_OPTIONS
0x20C3	PMERR_INV_RGBCOLOR
0x20C4	PMERR_INV_SCAN_START
0x20C5	PMERR_INV_SEG_ATTR
0x20C6	PMERR_INV_SEG_ATTR_VALUE
0x20C7	PMERR_INV_SEG_CH_LENGTH
0x20C8	PMERR_INV_SEG_NAME
0x20C9	PMERR_INV_SEG_OFFSET
0x20CA	PMERR_INV_SETID
0x20CB	PMERR_INV_SETID_TYPE
0x20CC	PMERR_INV_SET_VIEWPORT_OPTION
0x20CD	PMERR_INV_SHARPNESS_PARM
0x20CE	PMERR_INV_SOURCE_OFFSET
0x20CF	PMERR_INV_STOP_DRAW_VALUE
0x20D0	PMERR_INV_TRANSFORM_TYPE
0x20D1	PMERR_INV_USAGE_PARM
0x20D2	PMERR_INV_VIEWING_LIMITS
0x20D3	PMERR_JFILE_BUSY
0x20D4	PMERR_JNL_FUNC_DATA_TOO_LONG
0x20D5	PMERR_KERNING_NOT_SUPPORTED
0x20D6	PMERR_LABEL_NOT_FOUND
0x20D7	PMERR_MATRIX_OVERFLOW
0x20D8	PMERR_METAFILE_INTERNAL_ERROR
0x20D9	PMERR_METAFILE_IN_USE
0x20DA	PMERR_METAFILE_LIMIT_EXCEEDED
0x20DB	PMERR_NAME_STACK_FULL
0x20DC	PMERR_NOT_CREATED_BY_DEVOPENDC
0x20DD	PMERR_NOT_IN_AREA
0x20DE	PMERR_NOT_IN_DRAW_MODE
0x20DF	PMERR_NOT_IN_ELEMENT
0x20E0	PMERR_NOT_IN_IMAGE
0x20E1	PMERR_NOT_IN_PATH
0x20E2	PMERR_NOT_IN_RETAIN_MODE
0x20E3	PMERR_NOT_IN_SEG
0x20E4	PMERR_NO_BITMAP_SELECTED
0x20E5	PMERR_NO_CURRENT_ELEMENT
0x20E6	PMERR_NO_CURRENT_SEG
0x20E7	PMERR_NO_METAFILE_RECORD_HANDLE
0x20E8	PMERR_ORDER_TOO_BIG
0x20E9	PMERR_OTHER_SET_ID_REFS
0x20EA	PMERR_OVERRAN_SEG
0x20EB	PMERR_OWN_SET_ID_REFS
0x20EC	PMERR_PATH_INCOMPLETE
0x20ED	PMERR_PATH_LIMIT_EXCEEDED
0x20EE	PMERR_PATH_UNKNOWN
0x20EF	PMERR_PEL_IS_CLIPPED
0x20F0	PMERR_PEL_NOT_AVAILABLE
0x20F1	PMERR_PRIMITIVE_STACK_EMPTY
0x20F2	PMERR_PROLOG_ERROR
0x20F3	PMERR_PROLOG_SEG_ATTR_NOT_SET
0x20F4	PMERR_PS_BUSY
0x20F5	PMERR_PS_IS_ASSOCIATED
0x20F6	PMERR_RAM_JNL_FILE_TOO_SMALL
0x20F7	PMERR_REALIZE_NOT_SUPPORTED
0x20F8	PMERR_REGION_IS_CLIP_REGION
0x20F9	PMERR_RESOURCE_DEPLETION
0x20FA	PMERR_SEG_AND_REFSEG_ARE_SAME
0x20FB	PMERR_SEG_CALL_RECURSIVE
0x20FC	PMERR_SEG_CALL_STACK_EMPTY
0x20FD	PMERR_SEG_CALL_STACK_FULL
0x20FE	PMERR_SEG_IS_CURRENT
0x20FF	PMERR_SEG_NOT_CHAINED
0x2100	PMERR_SEG_NOT_FOUND
0x2101	PMERR_SEG_STORE_LIMIT_EXCEEDED
0x2102	PMERR_SETID_IN_USE
0x2103	PMERR_SETID_NOT_FOUND
0x2104	PMERR_STARTDOC_NOT_ISSUED
0x2105	PMERR_STOP_DRAW_OCCURRED
0x2106	PMERR_TOO_MANY_METAFILES_IN_USE
0x2107	PMERR_TRUNCATED_ORDER
0x2108	PMERR_UNCHAINED_SEG_ZERO_INV
0x2109	PMERR_UNSUPPORTED_ATTR

0x210A	PMERR_UNSUPPORTED_ATTR_VALUE
0x210B	PMERR_ENDDOC_NOT_ISSUED
0x210C	PMERR_PS_NOT_ASSOCIATED
0x210D	PMERR_INV_FLOOD_FILL_OPTIONS
0x210E	PMERR_INV_FACENAME
0x210F	PMERR_PALETTE_SELECTED
0x2110	PMERR_NO_PALETTE_SELECTED
0x2111	PMERR_INV_HPAL
0x2112	PMERR_PALETTE_BUSY
0x2113	PMERR_START_POINT_CLIPPED
0x2114	PMERR_NO_FILL
0x2115	PMERR_INV_FACENAMEDESC
0x2116	PMERR_INV_BITMAP_DATA
0x2117	PMERR_INV_CHAR_ALIGN_ATTR
0x2118	PMERR_INV_HFONT
0x2119	PMERR_HFONT_IS_SELECTED
0x2120	PMERR_DRVR_NOT_SUPPORTED
0x2120	PMERR_RASTER_FONT
0x3001	HMERR_DDF_MEMORY
0x3002	HMERR_DDF_ALIGN_TYPE
0x3003	HMERR_DDF_BACKCOLOR
0x3004	HMERR_DDF_FORECOLOR
0x3005	HMERR_DDF_FONTSTYLE
0x3006	HMERR_DDF_REFTYPE
0x3007	HMERR_DDF_LIST_UNCLOSED
0x3008	HMERR_DDF_LIST_UNINITIALIZED
0x3009	HMERR_DDF_LIST_BREAKTYPE
0x300A	HMERR_DDF_LIST_SPACING
0x300B	HMERR_DDF_HINSTANCE
0x300C	HMERR_DDF_EXCEED_MAX_LENGTH
0x300D	HMERR_DDF_EXCEED_MAX_INC
0x300E	HMERR_DDF_INVALID_DDF
0x300F	HMERR_DDF_FORMAT_TYPE
0x3010	HMERR_DDF_INVALID_PARM
0x3011	HMERR_DDF_INVALID_FONT
0x3012	HMERR_DDF_SEVERE
0x4001	PMERR_SPL_DRIVER_ERROR
0x4001	MERR_SPL_DRIVER_ERROR
0x4002	PMERR_SPL_DEVICE_ERROR
0x4002	MERR_SPL_DEVICE_ERROR
0x4003	PMERR_SPL_DEVICE_NOT_INSTALLED
0x4003	MERR_SPL_DEVICE_NOT_INSTALLED
0x4004	PMERR_SPL_QUEUE_ERROR
0x4004	MERR_SPL_QUEUE_ERROR
0x4005	PMERR_SPL_INV_HSPL
0x4005	MERR_SPL_INV_HSPL
0x4006	PMERR_SPL_NO_DISK_SPACE
0x4006	MERR_SPL_NO_DISK_SPACE
0x4007	PMERR_SPL_NO_MEMORY
0x4007	MERR_SPL_NO_MEMORY
0x4008	PMERR_SPL_PRINT_ABORT
0x4008	MERR_SPL_PRINT_ABORT
0x4009	PMERR_SPL_SPOOLER_NOT_INSTALLED
0x4009	MERR_SPL_SPOOLER_NOT_INSTALLED
0x400A	PMERR_SPL_INV_FORMS_CODE
0x400A	MERR_SPL_INV_FORMS_CODE
0x400B	PMERR_SPL_INV_PRIORITY
0x400B	MERR_SPL_INV_PRIORITY
0x400C	PMERR_SPL_NO_FREE_JOB_ID
0x400C	MERR_SPL_NO_FREE_JOB_ID
0x400D	PMERR_SPL_NO_DATA
0x400D	MERR_SPL_NO_DATA
0x400E	PMERR_SPL_INV_TOKEN
0x400E	MERR_SPL_INV_TOKEN
0x400F	PMERR_SPL_INV_DATATYPE
0x400F	MERR_SPL_INV_DATATYPE
0x4010	PMERR_SPL_PROCESSOR_ERROR
0x4010	MERR_SPL_PROCESSOR_ERROR
0x4011	PMERR_SPL_INV_JOB_ID
0x4011	MERR_SPL_INV_JOB_ID
0x4012	PMERR_SPL_JOB_NOT_PRINTING
0x4012	MERR_SPL_JOB_NOT_PRINTING
0x4013	PMERR_SPL_JOB_PRINTING

0x4013	MERR_SPL_JOB_PRINTING
0x4014	PMERR_SPL_QUEUE_ALREADY_EXISTS
0x4014	MERR_SPL_QUEUE_ALREADY_EXISTS
0x4015	PMERR_SPL_INV_QUEUE_NAME
0x4015	MERR_SPL_INV_QUEUE_NAME
0x4016	PMERR_SPL_QUEUE_NOT_EMPTY
0x4016	MERR_SPL_QUEUE_NOT_EMPTY
0x4017	PMERR_SPL_DEVICE_ALREADY_EXISTS
0x4017	MERR_SPL_DEVICE_ALREADY_EXISTS
0x4018	PMERR_SPL_DEVICE_LIMIT_REACHED
0x4018	MERR_SPL_DEVICE_LIMIT_REACHED
0x4019	PMERR_SPL_STATUS_STRING_TRUNC
0x4019	MERR_SPL_STATUS_STRING_TRUNC
0x401A	PMERR_SPL_INV_LENGTH_OR_COUNT
0x401A	MERR_SPL_INV_LENGTH_OR_COUNT
0x401B	PMERR_SPL_FILE_NOT_FOUND
0x401B	MERR_SPL_FILE_NOT_FOUND
0x401C	PMERR_SPL_CANNOT_OPEN_FILE
0x401C	MERR_SPL_CANNOT_OPEN_FILE
0x401D	PMERR_SPL_DRIVER_NOT_INSTALLED
0x401D	MERR_SPL_DRIVER_NOT_INSTALLED
0x401E	PMERR_SPL_INV_PROCESSOR_DATTYPE
0x401E	MERR_SPL_INV_PROCESSOR_DATTYPE
0x401F	PMERR_SPL_INV_DRIVER_DATATYPE
0x401F	MERR_SPL_INV_DRIVER_DATATYPE
0x4020	PMERR_SPL_PROCESSOR_NOT_INST
0x4020	MERR_SPL_PROCESSOR_NOT_INST
0x4021	PMERR_SPL_NO_SUCH_LOG_ADDRESS
0x4021	MERR_SPL_NO_SUCH_LOG_ADDRESS
0x4022	PMERR_SPL_PRINTER_NOT_FOUND
0x4022	MERR_SPL_PRINTER_NOT_FOUND
0x4023	PMERR_SPL_DD_NOT_FOUND
0x4023	MERR_SPL_DD_NOT_FOUND
0x4024	PMERR_SPL_QUEUE_NOT_FOUND
0x4024	MERR_SPL_QUEUE_NOT_FOUND
0x4025	PMERR_SPL_MANY_QUEUES_ASSOC
0x4025	MERR_SPL_MANY_QUEUES_ASSOC
0x4026	PMERR_SPL_NO_QUEUES_ASSOCIATED
0x4026	MERR_SPL_NO_QUEUES_ASSOCIATED
0x4027	PMERR_SPL_INI_FILE_ERROR
0x4027	MERR_SPL_INI_FILE_ERROR
0x4028	PMERR_SPL_NO_DEFAULT_QUEUE
0x4028	MERR_SPL_NO_DEFAULT_QUEUE
0x4029	PMERR_SPL_NO_CURRENT_FORMS_CODE
0x4029	MERR_SPL_NO_CURRENT_FORMS_CODE
0x402A	PMERR_SPL_NOT_AUTHORISED
0x402A	MERR_SPL_NOT_AUTHORISED
0x402B	PMERR_SPL_TEMP_NETWORK_ERROR
0x402B	MERR_SPL_TEMP_NETWORK_ERROR
0x402C	PMERR_SPL_HARD_NETWORK_ERROR
0x402C	MERR_SPL_HARD_NETWORK_ERROR
0x402D	PMERR_DEL_NOT_ALLOWED
0x402D	MERR_DEL_NOT_ALLOWED
0x402E	PMERR_CANNOT_DEL_QP_REF
0x402E	MERR_CANNOT_DEL_QP_REF
0x402F	PMERR_CANNOT_DEL_QNAME_REF
0x402F	MERR_CANNOT_DEL_QNAME_REF
0x4030	PMERR_CANNOT_DEL_PRINTER_DD_REF
0x4030	MERR_CANNOT_DEL_PRINTER_DD_REF
0x4031	PMERR_CANNOT_DEL_PRN_NAME_REF
0x4031	MERR_CANNOT_DEL_PRN_NAME_REF
0x4032	PMERR_CANNOT_DEL_PRN_ADDR_REF
0x4032	MERR_CANNOT_DEL_PRN_ADDR_REF
0x4033	PMERR_SPOOLER_QP_NOT_DEFINED
0x4033	MERR_SPOOLER_QP_NOT_DEFINED
0x4034	PMERR_PRN_NAME_NOT_DEFINED
0x4034	MERR_PRN_NAME_NOT_DEFINED
0x4035	PMERR_PRN_ADDR_NOT_DEFINED
0x4035	MERR_PRN_ADDR_NOT_DEFINED
0x4036	PMERR_PRINTER_DD_NOT_DEFINED
0x4036	MERR_PRINTER_DD_NOT_DEFINED
0x4037	PMERR_PRINTER_QUEUE_NOT_DEFINED
0x4037	MERR_PRINTER_QUEUE_NOT_DEFINED

0x4038	PMERR_PRN_ADDR_IN_USE
0x4038	MERR_PRN_ADDR_IN_USE
0x4039	PMERR_SPL_TOO_MANY_OPEN_FILES
0x4039	MERR_SPL_TOO_MANY_OPEN_FILES
0x403A	PMERR_SPL_CP_NOT_REQD
0x403A	MERR_SPL_CP_NOT_REQD
0x4040	PMERR_UNABLE_TO_CLOSE_DEVICE
0x4040	MERR_UNABLE_TO_CLOSE_DEVICE
0x4FC9	PMERR_SPLMSGBOX_INFO_CAPTION
0x4FC9	MERR_SPLMSGBOX_INFO_CAPTION
0x4FCA	PMERR_SPLMSGBOX_WARNING_CAPTION
0x4FCA	MERR_SPLMSGBOX_WARNING_CAPTION
0x4FCB	PMERR_SPLMSGBOX_ERROR_CAPTION
0x4FCB	MERR_SPLMSGBOX_ERROR_CAPTION
0x4FCC	PMERR_SPLMSGBOX_SEVERE_CAPTION
0x4FCC	MERR_SPLMSGBOX_SEVERE_CAPTION
0x4FCD	PMERR_SPLMSGBOX_JOB_DETAILS
0x4FCD	MERR_SPLMSGBOX_JOB_DETAILS
0x4FCE	PMERR_SPLMSGBOX_ERROR_ACTION
0x4FCE	MERR_SPLMSGBOX_ERROR_ACTION
0x4FCF	PMERR_SPLMSGBOX_SEVERE_ACTION
0x4FCF	MERR_SPLMSGBOX_SEVERE_ACTION
0x4FD0	PMERR_SPLMSGBOX_BIT_0_TEXT
0x4FD0	MERR_SPLMSGBOX_BIT_0_TEXT
0x4FD1	PMERR_SPLMSGBOX_BIT_1_TEXT
0x4FD1	MERR_SPLMSGBOX_BIT_1_TEXT
0x4FD2	PMERR_SPLMSGBOX_BIT_2_TEXT
0x4FD2	MERR_SPLMSGBOX_BIT_2_TEXT
0x4FD3	PMERR_SPLMSGBOX_BIT_3_TEXT
0x4FD3	MERR_SPLMSGBOX_BIT_3_TEXT
0x4FD4	PMERR_SPLMSGBOX_BIT_4_TEXT
0x4FD4	MERR_SPLMSGBOX_BIT_4_TEXT
0x4FD5	PMERR_SPLMSGBOX_BIT_5_TEXT
0x4FD5	MERR_SPLMSGBOX_BIT_5_TEXT
0x4FD6	PMERR_SPLMSGBOX_BIT_15_TEXT
0x4FD6	MERR_SPLMSGBOX_BIT_15_TEXT
0x4FD7	PMERR_SPL_NOPATHBUFFER
0x4FD7	MERR_SPL_NOPATHBUFFER
0x4FD8	MERR_SPL_ALREADY_INITIALISED
0x4FD9	MERR_SPL_ERROR
0x5001	PMERR_INV_TYPE
0x5001	MERR_INV_TYPE
0x5002	PMERR_INV_CONV
0x5002	MERR_INV_CONV
0x5003	PMERR_INV_SEGLEN
0x5003	MERR_INV_SEGLEN
0x5004	PMERR_DUP_SEGNAME
0x5004	MERR_DUP_SEGNAME
0x5005	PMERR_INV_XFORM
0x5005	MERR_INV_XFORM
0x5006	PMERR_INV_VIEWLIM
0x5006	MERR_INV_VIEWLIM
0x5007	PMERR_INV_3DCOORD
0x5007	MERR_INV_3DCOORD
0x5008	PMERR_SMB_OVFLOW
0x5008	MERR_SMB_OVFLOW
0x5009	PMERR_SEG_OVFLOW
0x5009	MERR_SEG_OVFLOW
0x5010	PMERR_PIC_DUP_FILENAME
0x5010	MERR_PIC_DUP_FILENAME
SPLERR_BASE+0FA1	PMERR_SPL_ERROR_1
SPLERR_BASE+0FA2	PMERR_SPL_ERROR_2
SPLERR_BASE+0FA3	PMERR_SPL_ERROR_3
SPLERR_BASE+0FA4	PMERR_SPL_ERROR_4
SPLERR_BASE+0FA5	PMERR_SPL_ERROR_5
SPLERR_BASE+0FA6	PMERR_SPL_ERROR_6
SPLERR_BASE+0FA7	PMERR_SPL_ERROR_7
SPLERR_BASE+0FA8	PMERR_SPL_ERROR_8
SPLERR_BASE+0FA9	PMERR_SPL_ERROR_9
SPLERR_BASE+0FAA	PMERR_SPL_ERROR_10
SPLERR_BASE+0FAB	PMERR_SPL_ERROR_11
SPLERR_BASE+0FAC	PMERR_SPL_ERROR_12
SPLERR_BASE+0FAD	PMERR_SPL_ERROR_13

SPLERR_BASE+0FAE	PMERR_SPL_ERROR_14
SPLERR_BASE+0FAF	PMERR_SPL_ERROR_15
SPLERR_BASE+0FB0	PMERR_SPL_ERROR_16
SPLERR_BASE+0FB1	PMERR_SPL_ERROR_17
SPLERR_BASE+0FB2	PMERR_SPL_ERROR_18
SPLERR_BASE+0FB3	PMERR_SPL_ERROR_19
SPLERR_BASE+0FB4	PMERR_SPL_ERROR_20
SPLERR_BASE+0FB5	PMERR_SPL_ERROR_21
SPLERR_BASE+0FB6	PMERR_SPL_ERROR_22
SPLERR_BASE+0FB7	PMERR_SPL_ERROR_23
SPLERR_BASE+0FB8	PMERR_SPL_ERROR_24
SPLERR_BASE+0FB9	PMERR_SPL_ERROR_25
SPLERR_BASE+0FBA	PMERR_SPL_ERROR_26
SPLERR_BASE+0FBB	PMERR_SPL_ERROR_27
SPLERR_BASE+0FBC	PMERR_SPL_ERROR_28
SPLERR_BASE+0FBD	PMERR_SPL_ERROR_29
SPLERR_BASE+0FBE	PMERR_SPL_ERROR_30
SPLERR_BASE+0FBF	PMERR_SPL_ERROR_31
SPLERR_BASE+0FC0	PMERR_SPL_ERROR_32
SPLERR_BASE+0FC1	PMERR_SPL_ERROR_33
SPLERR_BASE+0FC2	PMERR_SPL_ERROR_34
SPLERR_BASE+0FC3	PMERR_SPL_ERROR_35
SPLERR_BASE+0FC4	PMERR_SPL_ERROR_36
SPLERR_BASE+0FC5	PMERR_SPL_ERROR_37
SPLERR_BASE+0FC6	PMERR_SPL_ERROR_38
SPLERR_BASE+0FC7	PMERR_SPL_ERROR_39
SPLERR_BASE+0FC8	PMERR_SPL_ERROR_40
SPLERR_BASE+0FFF	PMERR_SPL_ERROR
SPLERR_BASE+0FFD	PMERR_SPL_ALREADY_INITIALISED

Error Name and Explanation

This appendix gives an explanation for each PM error. The errors are listed in alphabetic order. The number associated with each error is given in Error Number and Name.

Error Constant	Explanation
HMERR_ALLOCATE_SEGMENT	Unable to allocate a segment of memory for memory allocation requests from the Help Manager.
HMERR_CLOSE_LIB_FILE	The library file cannot be closed.
HMERR_CONTENT_NOT_FOUND	The library file does not have any content.
HMERR_DATABASE_NOT_OPEN	Unable to read the unopened database.
HMERR_DDF_ALIGN_TYPE	The alignment type is not valid.
HMERR_DDF_BACKCOLOR	The background color is not valid.
HMERR_DDF_EXCEED_MAX_INC	The value specified to increment DDF memory is too large.
HMERR_DDF_EXCEED_MAX_LENGTH	The amount of data is too large for the DDF buffer.
HMERR_DDF_FONTSTYLE	The font style is not valid.

HMERR_DDF_FORECOLOR	The foreground color is not valid.
HMERR_DDF_FORMAT_TYPE	The format type specified is invalid.
HMERR_DDF_HINSTANCE	The DDF instance is invalid.
HMERR_DDF_INVALID_DDF	The DDF handle is invalid.
HMERR_DDF_INVALID_FONT	The font value specified is invalid.
HMERR_DDF_INVALID_PARM	One of the DDF parameters specified is invalid.
HMERR_DDF_LIST_BREAKTYPE	The value of BreakType is not valid.
HMERR_DDF_LIST_SPACING	The value for Spacing is not valid.
HMERR_DDF_LIST_UNCLOSED	An attempt was made to nest a list.
HMERR_DDF_LIST_UNINITIALIZED	No definition list has been initialized by DdfBeginList.
HMERR_DDF_MEMORY	Not enough memory is available.
HMERR_DDF_REFTYPE	The reference type is not valid.
HMERR_DDF_SEVERE	Internal error detected by the Help Manager.
HMERR_FREE_MEMORY	Unable to free allocated memory.
HMERR_HELP_INST_CALLED_INVALID	The handle of the instance specified on a call to the Help Manager does not have the class name of a Help Manager instance.
HMERR_HELP_INSTANCE_UNDEFINE	The help instance handle specified is invalid.
HMERR_HELPITEM_NOT_FOUND	Context-sensitive help was requested but the ID of the main help item specified was not found in the help table.
HMERR_HELPSUBITEM_NOT_FOUND	Context-sensitive help was requested but the ID of the help item specified was not found in the help subtable.
HMERR_HELPTABLE_UNDEFINE	The application did not provide a help table for context-sensitive help.
HMERR_INDEX_NOT_FOUND	The index is not in the library file.
HMERR_INVALID_ASSOC_APP_WND	The application window handle specified on the WinAssociateHelpInstance function is not a valid window handle.
HMERR_INVALID_ASSOC_HELP_INST	The help instance handle

	specified on the WinAssociateHelpInstance function is not a valid window handle.
HMERR_INVALID_DESTROY_HELP_INST	The window handle specified as the help instance to destroy is not of the help instance class.
HMERR_INVALID_HELP_INSTANCE_HDL	The handle specified to be a help instance does not have the class name of a Help Manager instance.
HMERR_INVALID_HELPSUBITEM_SIZE	The help subtable item size is less than 2.
HMERR_INVALID_LIB_FILE	Improper library file provided.
HMERR_INVALID_QUERY_APP_WND	The application window specified on a WinQueryHelpInstance function is not a valid window handle.
HMERR_LOAD_DLL	Unable to load resource data link library.
HMERR_NO_FRAME_WND_IN_CHAIN	There is no frame window in the window chain from which to find or set the associated help instance.
HMERR_NO_HELP_INST_IN_CHAIN	The parent or owner chain of the application window specified does not have an associated help instance.
HMERR_NO_MEMORY	Unable to allocate the requested amount of memory.
HMERR_OPEN_LIB_FILE	The library file cannot be opened.
HMERR_PANEL_NOT_FOUND	Unable to find the requested help panel.
HMERR_READ_LIB_FILE	The library file cannot be read.
PMERR_ACCESS_DENIED	The memory block was not allocated properly.
PMERR_ALREADY_IN_AREA	An attempt was made to begin a new area while an existing area bracket was already open.
PMERR_ALREADY_IN_ELEMENT	An attempt was made to begin a new element while an existing element bracket was already open.
PMERR_ALREADY_IN_PATH	An attempt was made to begin a new path while an existing path bracket was already open.
PMERR_ALREADY_IN_SEG	An attempt was made to open a new segment while an existing segment bracket

	was already open.
PMERR_APPL_STRUCTURE_TOO_SMALL	The application buffer length is less than the total length required for the (application) component types.
PMERR_AREA_INCOMPLETE	One of the following has occurred: o A segment has been opened , closed, or drawn. o GpiAssociate was issued while an area bracket was open. o A drawn segment has opened an area bracket and ended without closing it.
PMERR_ARRAY_TOO_LARGE	More than 4 bytes was attempted to be inserted or extracted.
PMERR_ARRAY_TOO_SMALL	The array specified was too small.
PMERR_ATOM_NAME_NOT_FOUND	The specified atom name is not in the atom table.
PMERR_BASE_ERROR	An OS/2 base error has occurred. The base error code can be accessed using the OffBinaryData field of the ERRINFO structure returned by WinGetErrorInfo.
PMERR_BITMAP_IN_USE	An attempt was made either to set a bit map into a device context using GpiSetBitmap while it was already selected into an existing device context, or to tag a bit map with a local pattern set identifier (setid) using GpiSetBitmapId while it was already tagged with an existing setid.
PMERR_BITMAP_IS_SELECTED	An attempt was made to delete a bit map while it was selected into a device context.
PMERR_BITMAP_NOT_FOUND	A attempt was made to perform a bit-map operation on a bit map that did not exist.
PMERR_BITMAP_NOT_SELECTED	A attempt was made to perform an operation on presentation space associated with a memory device context that had no selected bit map.
PMERR_BOUNDS_OVERFLOW	An internal overflow error occurred during boundary data accumulation. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.
PMERR_BUFFER_TOO_SMALL	The supplied buffer was not large enough for the data to be returned.

PMERR_C_LENGTH_TOO_SMALL	The maximum length of the C structure is less than the total length required for the (C) component types.
PMERR_CALLED_SEG_IS_CHAINED	An attempt was made to call a segment that has a chained attribute set.
PMERR_CALLED_SEG_IS_CURRENT	An attempt was made to call a segment that is currently open.
PMERR_CALLED_SEG_NOT_FOUND	An attempt was made to call a segment that did not exist.
PMERR_CAN_NOT_CALL_SPOOLER	An error occurred attempting to call the spooler validation routine. This error is not raised if the spooler is not installed.
PMERR_CANNOT_DEL_PRINTER_DD_REF	Presentation Manager device driver deletion not possible due to a reference.
PMERR_CANNOT_DEL_PRN_ADDR_REF	Printer port deletion not possible due to a reference.
PMERR_CANNOT_DEL_PRN_NAME_REF	Printer deletion not possible due to a reference.
PMERR_CANNOT_DEL_QNAME_REF	Spooler queue deletion not possible due to a reference.
PMERR_CANNOT_DEL_QP_REF	Spooler queue processor deletion not possible due to a reference.
PMERR_CANNOT_SET_FOCUS	Focus cannot be set if a focus change is in progress, or if a system-modal window exists.
PMERR_CANNOT_STOP	The session cannot be stopped.
PMERR_COL_TABLE_NOT_REALIZABLE	An attempt was made to realize a color table that is not realizable.
PMERR_COL_TABLE_NOT_REALIZED	An attempt was made to realize a color table on a device driver that does not support this function.
PMERR_COORDINATE_OVERFLOW	An internal coordinate overflow error occurred. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.
PMERR_DATA_TOO_LONG	An attempt was made to transfer more than the maximum permitted amount of data (64512 bytes) using GpiPutData, GpiGetData, or GpiElement.
PMERR_DATATYPE_ENTRY_BAD_INDEX	An invalid datatype entry index was specified.

PMERR_DATATYPE_ENTRY_CTL_BAD	An invalid datatype entry control was specified.
PMERR_DATATYPE_ENTRY_CTL_MISS	The datatype entry control was missing.
PMERR_DATATYPE_ENTRY_NOT_NUM	The datatype entry specified was not numerical.
PMERR_DATATYPE_ENTRY_NOT_OFF	The datatype entry specified was not an offset.
PMERR_DATATYPE_INVALID	An invalid datatype was specified.
PMERR_DATATYPE_NOT_UNIQUE	An attempt to register a datatype failed because it is not unique.
PMERR_DATATYPE_TOO_LONG	The datatype specified was too long.
PMERR_DATATYPE_TOO_SMALL	The datatype specified was too small.
PMERR_DC_IS_ASSOCIATED	An attempt was made to associate a presentation space with a device context that was already associated or to destroy a device context that was associated.
PMERR_DEL_NOT_ALLOWED	Deletion not possible.
PMERR_DESC_STRING_TRUNCATED	An attempt was made to supply a description string with GpiBeginElement that was greater then the permitted maximum length (251 characters). The string was truncated.
PMERR_DEV_FUNC_NOT_INSTALLED	The function requested is not supported by the presentation driver.
PMERR_DEVICE_DRIVER_ERROR_1	Miscellaneous error available for use by user written device drivers.
PMERR_DEVICE_DRIVER_ERROR_10	Miscellaneous error available for use by user written device drivers.
PMERR_DEVICE_DRIVER_ERROR_2	Miscellaneous error available for use by user written device drivers.
PMERR_DEVICE_DRIVER_ERROR_3	Miscellaneous error available for use by user written device drivers.
PMERR_DEVICE_DRIVER_ERROR_4	Miscellaneous error available for use by user written device drivers.
PMERR_DEVICE_DRIVER_ERROR_5	Miscellaneous error available for use by user written device drivers.
PMERR_DEVICE_DRIVER_ERROR_6	Miscellaneous error available for use by user written device drivers.

PMERR_DEVICE_DRIVER_ERROR_7	Miscellaneous error available for use by user written device drivers.
PMERR_DEVICE_DRIVER_ERROR_8	Miscellaneous error available for use by user written device drivers.
PMERR_DEVICE_DRIVER_ERROR_9	Miscellaneous error available for use by user written device drivers.
PMERR_DOS_ERROR	A DOS call returned an error.
PMERR_DOSOPEN_FAILURE	A DosOpen call made during GpiLoadMetaFile or GpiSaveMetaFile gave a good return code but the file was not opened successfully.
PMERR_DOSREAD_FAILURE	A DosRead call made during GpiLoadMetaFile gave a good return code. However, it failed to read any more bytes although the file length indicated that there were more to be read.
PMERR_DRIVER_NOT_FOUND	The device driver specified with DevPostDeviceModes was not found.
PMERR_DUP_SEG	During GpiPlayMetaFile, while the actual drawing mode was **draw-and-retain** or **retain**, a metafile segment to be stored in the presentation space was found to have the same segment identifier as an existing segment.
PMERR_DUP_SEGNAME	A called segment has a name that has already been used by another called segment in the input PIF.
PMERR_DUPLICATE_TITLE	The program title specified in the PIBSTRUCT already exists within the same group.
PMERR_DYNAMIC_SEG_SEQ_ERROR	During removal of dynamic segments while processing GpiDrawChain, GpiDrawFrom, or GpiDrawSegment, the internal state indicated that dynamic segment data was still visible after all chained dynamic segments had been processed. This can occur if segments drawn dynamically (including called segments) are modified or removed from the chain while visible.
PMERR_DYNAMIC_SEG_ZERO_INV	An attempt was been made to open a dynamic segment with a segment identifier of zero.
PMERR_ENDDOC_NOT_ISSUED	A request to close the

	spooled output without first issuing a an ENDDOC was attempted.
PMERR_ESC_CODE_NOT_SUPPORTED	The code specified with DevEscape is not supported by the target device driver.
PMERR_EXCEEDS_MAX_SEG_LENGTH	During metafile creation or generation of retained graphics the system has exceeded maximum segment size.
PMERR_FONT_AND_MODE_MISMATCH	An attempt was made to draw characters with a character mode and character set that are incompatible. For example, the character specifies an image/raster font when the mode calls for a vector/outline font.
PMERR_FONT_FILE_NOT_LOADED	An attempt was made to unload a font file that was not loaded.
PMERR_FONT_NOT_LOADED	An attempt was made to create a font that was not loaded.
PMERR_FUNCTION_NOT_SUPPORTED	The function is not supported.
PMERR_GREATER_THAN_64K	A data item or array dimension is greater than 65 535.
PMERR_HBITMAP_BUSY	An internal bit map busy error was detected. The bit map was locked by one thread during an attempt to access it from another thread.
PMERR_HDC_BUSY	An internal device context busy error was detected. The device context was locked by one thread during an attempt to access it from another thread.
PMERR_HEAP_MAX_SIZE_REACHED	The heap has reached its maximum size (64KB), and cannot be increased.
PMERR_HEAP_OUT_OF_MEMORY	An attempt to increase the size of the heap failed.
PMERR_HFONT_IS_SELECTED	An attempt has been made to either change the owner of a font, or delete when it is currently selected.
PMERR_HRGN_BUSY	An internal region busy error was detected. The region was locked by one thread during an attempt to access it from another thread.
PMERR_HUGE_FONTS_NOT_SUPPORTED	An attempt was made using GpiSetCharSet, GpiSetPatternSet,

	GpiSetMarkerSet, or GpiSetAttrs to select a font that is larger than the maximum size (64Kb) supported by the target device driver.
PMERR_ID_HAS_NO_BITMAP	No bit map was tagged with the setid specified on a GpiQueryBitmapHandle function.
PMERR_IMAGE_INCOMPLETE	A drawn segment has opened an image bracket and ended without closing it.
PMERR_INCOMPATIBLE_BITMAP	An attempt was made to select a bit map or perform a BitBlt operation on a device context that was incompatible with the format of the bit map.
PMERR_INCOMPATIBLE_METAFILE	An attempt was made to associate a presentation space and a metafile device context with incompatible page units, size or coordinate format; or to play a metafile using the RES_RESET option (to reset the presentation space) to a presentation space that is itself associated with a metafile device context.
PMERR_INCORRECT_DATATYPE	A data type is specified which is incorrect for this function.
PMERR_INCORRECT_DC_TYPE	An attempt was made to perform a bit-map operation on a presentation space associated with a device context of a type that is unable to support bit-map operations.
PMERR_INCORRECT_HSTRUCT	A structure handle is non-NULL, and is invalid for one of the following reasons: o It is not the handle of a data structure. o It is the handle of an ERRINFO structure, which should not be used in this call. o A handle block returned by the bindin to the application has been used for ar in-line structure handle.
PMERR_INI_FILE_IS_SYS_OR_USER	User or system initialization file cannot be closed.
PMERR_INSUFF_SPACE_TO_ADD	The initialization file could not be extended to add the required program or group.
PMERR_INSUFFICIENT_DISK_SPACE	The operation terminated through insufficient disk space.
PMERR_INSUFFICIENT_MEMORY	The operation terminated through insufficient memory.
PMERR_INV_3DCOORD	An order specifying

	3-dimensional coordinates has been found in the input PIF.
PMERR_INV_ANGLE_PARM	An invalid angle parameter was specified with GpiPartialArc.
PMERR_INV_ARC_CONTROL	An invalid control parameter was specified with GpiFullArc.
PMERR_INV_AREA_CONTROL	An invalid options parameter was specified with GpiBeginArea.
PMERR_INV_ATTR_MODE	An invalid mode parameter was specified with GpiSetAttrMode.
PMERR_INV_BACKGROUND_COL_ATTR	An invalid background color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_BACKGROUND_MIX_ATTR	An invalid background mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_BITBLT_MIX	An invalid *lRop* was specified with a GpiBitBlt or GpiWCBitBlt function.
PMERR_INV_BITBLT_STYLE	An invalid options parameter was specified with a GpiBitBlt or GpiWCBitBlt function.
PMERR_INV_BITMAP_DATA	In processing a bit map, the end of the data was unexpectedly encountered.
PMERR_INV_BITMAP_DIMENSION	An invalid dimension was specified with a load bit-map function.
PMERR_INV_BOX_CONTROL	An invalid control parameter was specified with GpiBox.
PMERR_INV_BOX_ROUNDING_PARM	An invalid corner rounding control parameter was specified with GpiBox.
PMERR_INV_CHAR_ALIGN_ATTR	The text alignment attribute specified in GpiSetTextAlignment is not valid.
PMERR_INV_CHAR_ANGLE_ATTR	The default character angle attribute value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_DIRECTION_ATTR	An invalid character direction attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using

	the defaults mask.
PMERR_INV_CHAR_MODE_ATTR	An invalid character mode attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_POS_OPTIONS	An invalid options parameter was specified with GpiCharStringPos or GpiCharStringPosAt.
PMERR_INV_CHAR_SET_ATTR	An invalid character setid attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CHAR_SHEAR_ATTR	An invalid character shear attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_CLIP_PATH_OPTIONS	An invalid options parameter was specified with GpiSetClipPath.
PMERR_INV_CODEPAGE	An invalid code-page parameter was specified with GpiSetCp.
PMERR_INV_COLOR_ATTR	An invalid color attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_COLOR_DATA	Invalid color table definition data was specified with GpiCreateLogColorTable.
PMERR_INV_COLOR_FORMAT	An invalid format parameter was specified with GpiCreateLogColorTable.
PMERR_INV_COLOR_INDEX	An invalid color index parameter was specified with GpiQueryRGBColor.
PMERR_INV_COLOR_OPTIONS	An invalid options parameter was specified with a logical color table or color query function.
PMERR_INV_COLOR_START_INDEX	An invalid starting index parameter was specified with a logical color table or color query function.
PMERR_INV_CONV	Invalid conversion-type parameter.
PMERR_INV_COORD_OFFSET	An invalid coordinate offset value was specified.
PMERR_INV_COORD_SPACE	An invalid source or target coordinate space parameter

	was specified with GpiConvert.
PMERR_INV_COORDINATE	An invalid coordinate value was specified.
PMERR_INV_CORRELATE_DEPTH	An invalid maxdepth parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_CORRELATE_TYPE	An invalid type parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_CURSOR_BITMAP	An invalid pointer was referenced with WinSetPointer.
PMERR_INV_DC_DATA	An invalid data parameter was specified with DevOpenDC.
PMERR_INV_DC_TYPE	An invalid type parameter was specified with DevOpenDC, or a function was issued that is invalid for a OD_METAFILE_NOQUERY device context.
PMERR_INV_DEV_MODES_OPTIONS	An invalid options parameter was specified with DevPostDeviceModes.
PMERR_INV_DEVICE_NAME	An invalid devicename parameter was specified with DevPostDeviceModes.
PMERR_INV_DRAW_BORDER_OPTION	An invalid option parameter was specified with WinDrawBorder.
PMERR_INV_DRAW_CONTROL	An invalid control parameter was specified with GpiSetDrawControl or GpiQueryDrawControl.
PMERR_INV_DRAW_VALUE	An invalid value parameter was specified with GpiSetDrawControl.
PMERR_INV_DRAWING_MODE	An invalid mode parameter was specified with GpiSetDrawControl not **draw-and-retain** or **draw**.
PMERR_INV_DRIVER_DATA	Invalid driver data was specified.
PMERR_INV_DRIVER_NAME	A driver name was specified which has not been installed.
PMERR_INV_EDIT_MODE	An invalid mode parameter was specified with GpiSetEditMode.
PMERR_INV_ELEMENT_OFFSET	An invalid off (offset) parameter was specified with GpiQueryElement.

PMERR_INV_ELEMENT_POINTER	An attempt was made to issue GpiPutData with the element pointer not pointing at the last element.
PMERR_INV_END_PATH_OPTIONS	An attempt to create or delete a path out of context of the path bracket was made.
PMERR_INV_ESC_CODE	An invalid escape code was used in a call to DevEscape.
PMERR_INV_ESCAPE_DATA	An invalid data parameter was specified with DevEscape.
PMERR_INV_FACENAME	An invalid font family name was passed to GpiQueryFaceString.
PMERR_INV_FACENAMEDESC	The font facename description is invalid.
PMERR_INV_FILL_PATH_OPTIONS	An invalid options parameter was specified with GpiFillPath.
PMERR_INV_FIRST_CHAR	An invalid firstchar parameter was specified with GpiQueryWidthTable.
PMERR_INV_FLOOD_FILL_OPTIONS	Invalid flood fill parameters were specified.
PMERR_INV_FONT_ATTRS	An invalid attrs parameter was specified with GpiCreateLogFont.
PMERR_INV_FONT_FILE_DATA	The font file specified with GpiLoadFonts, GpiLoadPublicFonts, GpiQueryFontFileDescriptions, or GpiQueryFullFontFileDescs contains invalid data.
PMERR_INV_FOR_THIS_DC_TYPE	An attempt has been made to issue GpiRemoveDynamics or GpiDrawDynamics to a presentation space associated with a metafile device context.
PMERR_INV_FORMS_CODE	An invalid forms code parameter was specified with DevQueryHardcopyCaps.
PMERR_INV_GEOM_LINE_WIDTH_ATTR	An invalid geometric line width attribute value was specified.
PMERR_INV_GETDATA_CONTROL	An invalid format parameter was specified with GpiGetData.
PMERR_INV_GRAPHICS_FIELD	An invalid field parameter was specified with GpiSetGraphicsField.
PMERR_INV_HBITMAP	An invalid bit-map handle was specified.

PMERR_INV_HDC	An invalid device-context handle or (micro presentation space) presentation-space handle was specified.
PMERR_INV_HFONT	An invalid font handle was specified.
PMERR_INV_HMF	An invalid metafile handle was specified.
PMERR_INV_HPAL	An invalid color palette handle was specified.
PMERR_INV_HPS	An invalid presentation-space handle was specified.
PMERR_INV_HRGN	An invalid region handle was specified.
PMERR_INV_ID	An invalid *lPSid* parameter was specified with GpiRestorePS.
PMERR_INV_IMAGE_DATA_LENGTH	An invalid *lLength* parameter was specified with GpiImage. There is a mismatch between the image size and the data length.
PMERR_INV_IMAGE_DIMENSION	An invalid *psizlImageSize* parameter was specified with GpiImage.
PMERR_INV_IMAGE_FORMAT	An invalid *lFormat* parameter was specified with GpiImage.
PMERR_INV_IN_AREA	An attempt was made to issue a function invalid inside an area bracket. This can be detected while the actual drawing mode is **draw** or **draw-and-retain** or during segment drawing or correlation functions.
PMERR_INV_IN_CURRENT_EDIT_MODE	An attempt was made to issue a function invalid inside the current editing mode.
PMERR_INV_IN_ELEMENT	An attempt was made to issue a function invalid inside an element bracket.
PMERR_INV_IN_IMAGE	An attempt was made to issue a function invalid inside an element bracket.
PMERR_INV_IN_PATH	An attempt was made to issue a function invalid inside a path bracket.
PMERR_INV_IN_RETAIN_MODE	An attempt was made to issue a function (for example, query) that is invalid when the actual drawing mode is not **draw** or **draw-and-retain**.
PMERR_INV_IN_SEG	An attempt was made to issue a function invalid inside a segment bracket.

PMERR_INV_IN_VECTOR_SYMBOL	An invalid order was detected inside a vector symbol definition while drawing a vector (outline) font.
PMERR_INV_INFO_TABLE	An invalid bit-map info table was specified with a bit-map operation.
PMERR_INV_LENGTH_OR_COUNT	An invalid length or count parameter was specified.
PMERR_INV_LINE_END_ATTR	An invalid line end attribute value was specified.
PMERR_INV_LINE_JOIN_ATTR	An invalid line join attribute value was specified.
PMERR_INV_LINE_TYPE_ATTR	An invalid line type attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_LINE_WIDTH_ATTR	An invalid line width attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_LOGICAL_ADDRESS	An invalid device logical address was specified.
PMERR_INV_MARKER_BOX_ATTR	An invalid marker box attribute value was specified.
PMERR_INV_MARKER_SET_ATTR	An invalid marker set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_MARKER_SYMBOL_ATTR	An invalid marker symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_MATRIX_ELEMENT	An invalid transformation matrix element was specified.
PMERR_INV_MAX_HITS	An invalid maxhits parameter was specified with GpiCorrelateSegment, GpiCorrelateFrom, or GpiCorrelateChain.
PMERR_INV_METAFILE	An invalid metafile was specified with GpiPlayMetaFile.
PMERR_INV_METAFILE_LENGTH	An invalid length parameter was specified with GpiSetMetaFileBits or GpiQueryMetaFileBits.
PMERR_INV_METAFILE_OFFSET	An invalid length parameter was specified with

	GpiSetMetaFileBits or GpiQueryMetaFileBits.
PMERR_INV_MICROPS_DRAW_CONTROL	A draw control parameter was specified with GpiSetDrawControl that is invalid in a micro presentation space.
PMERR_INV_MICROPS_FUNCTION	An attempt was made to issue a function that is invalid in a micro presentation space.
PMERR_INV_MICROPS_ORDER	An attempt was made to play a metafile containing orders that are invalid in a micro presentation space.
PMERR_INV_MIX_ATTR	An invalid mix attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_MODE_FOR_OPEN_DYN	An attempt was made to open a segment with the ATTR_DYNAMIC segment set, while the drawing mode was set to DM_DRAW or DM_DRAWANDRETAIN.
PMERR_INV_MODE_FOR_REOPEN_SEG	An attempt was made to reopen an existing segment while the drawing mode was set to DM_DRAW or DM_DRAWANDRETAIN.
PMERR_INV_MODIFY_PATH_MODE	An invalid mode parameter was specified with GpiModifyPath.
PMERR_INV_MULTIPLIER	An invalid multiplier parameter was specified with GpiPartialArc or GpiFullArc.
PMERR_INV_NESTED_FIGURES	Nested figures have been detected within a path definition.
PMERR_INV_OR_INCOMPAT_OPTIONS	An invalid or incompatible (with micro presentation space) options parameter was specified with GpiCreatePS or GpiSetPS.
PMERR_INV_ORDER_LENGTH	An invalid order length was detected during GpiPutData or segment drawing.
PMERR_INV_ORDERING_PARM	An invalid order parameter was specified with GpiSetSegmentPriority.
PMERR_INV_OUTSIDE_DRAW_MODE	An attempt was made to issue a GpiSavePS or GpiRestorePS function, or an output only function (for example, GpiPaintRegion) from GpiPlayMetaFile without the drawing mode set to DM_DRAW.

PMERR_INV_PAGE_VIEWPORT	An invalid viewport parameter was specified with GpiSetPageViewport.
PMERR_INV_PATH_ID	An invalid path identifier parameter was specified.
PMERR_INV_PATTERN_ATTR	An invalid pattern symbol attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_PATTERN_REF_PT_ATTR	An invalid refpoint attribute value was specified.
PMERR_INV_PATTERN_SET_ATTR	An invalid pattern set attribute value was specified or the default value was explicitly specified with GpiSetAttrs instead of using the defaults mask.
PMERR_INV_PATTERN_SET_FONT	An attempt was made to use an unsuitable font as a pattern set.
PMERR_INV_PICK_APERTURE_OPTION	An invalid options parameter was specified with GpiSetPickApertureSize.
PMERR_INV_PICK_APERTURE_POSN	An invalid pick aperture position was specified.
PMERR_INV_PICK_APERTURE_SIZE	An invalid size parameter was specified with GpiSetPickApertureSize.
PMERR_INV_PLAY_METAFILE_OPTION	An invalid option parameter was specified with GpiPlayMetaFile.
PMERR_INV_PRIMITIVE_TYPE	An invalid primitive type parameter was specified with GpiSetAttrs or GpiQueryAttrs.
PMERR_INV_PS_SIZE	An invalid size parameter was specified with GpiCreatePS or GpiSetPS.
PMERR_INV_PUTDATA_FORMAT	An invalid format parameter was specified with GpiPutData.
PMERR_INV_QUERY_ELEMENT_NO	An invalid start parameter was specified with DevQueryCaps.
PMERR_INV_RECT	An invalid rectangle parameter was specified.
PMERR_INV_REGION_CONTROL	An invalid control parameter was specified with GpiQueryRegionRects.
PMERR_INV_REGION_MIX_MODE	An invalid mode parameter was specified with GpiCombineRegion.
PMERR_INV_REPLACE_MODE_FUNC	An attempt was made to issue GpiPutData with the

	editing mode set to SEGEM_REPLACE.
PMERR_INV_RESERVED_FIELD	An invalid reserved field was specified.
PMERR_INV_RESET_OPTIONS	An invalid options parameter was specified with GpiResetPS.
PMERR_INV_RGBCOLOR	An invalid rgb color parameter was specified with GpiQueryNearestColor or GpiQueryColor.
PMERR_INV_SCAN_START	An invalid scanstart parameter was specified with a bit-map function.
PMERR_INV_SEG_ATTR	An invalid attribute parameter was specified with GpiSetSegmentAttrs, GpiQuerySegmentAttrs, GpiSetInitialSegmentAttrs, or GpiQueryInitialSegmentAttrs.
PMERR_INV_SEG_ATTR_VALUE	An invalid attribute value parameter was specified with GpiSetSegmentAttrs or GpiSetInitialSegmentAttrs.
PMERR_INV_SEG_NAME	An invalid segment identifier was specified.
PMERR_INV_SEG_OFFSET	An invalid offset parameter was specified with GpiPutData.
PMERR_INV_SEGLEN	An order length exceeds the remaining segment length in the input PIF.
PMERR_INV_SETID	An invalid setid parameter was specified.
PMERR_INV_SHARPNESS_PARM	An invalid sharpness parameter was specified with GpiPolyFilletSharp.
PMERR_INV_STOP_DRAW_VALUE	An invalid value parameter was specified with GpiSetStopDraw.
PMERR_INV_TRANSFORM_TYPE	An invalid options parameter was specified with a transform matrix function.
PMERR_INV_TYPE	Invalid file-type parameter.
PMERR_INV_USAGE_PARM	An invalid options parameter was specified with GpiCreateBitmap.
PMERR_INV_VIEWING_LIMITS	An invalid limits parameter was specified with GpiSetViewingLimits.
PMERR_INV_VIEWLIM	A set viewing limits order has an inconsistent mask and order length in the input PIF.
PMERR_INV_XFORM	A set (default) viewing transform order has an

	inconsistent mask and order length in the input PIF.
PMERR_INVALID_APPL	Attempted to start an application whose type is not recognized by OS/2.
PMERR_INVALID_ARRAY_COUNT	An array has an invalid count, that is, less than or equal to zero.
PMERR_INVALID_ARRAY_SIZE	A control data type array size is invalid.
PMERR_INVALID_ASCIIZ	The profile string is not a valid zero-terminated string.
PMERR_INVALID_ATOM	The specified atom does not exist in the atom table.
PMERR_INVALID_ATOM_NAME	An invalid atom name string was passed.
PMERR_INVALID_BUNDLE_TYPE	An invalid bundle type was passed.
PMERR_INVALID_CHARACTER_INDEX	On WinNextChar or WinPrevChar, a character index is invalid, that is, it is less than 1 or is greater than the string length+1.
PMERR_INVALID_CONTROL_DATATYPE	An invalid control data type was specified.
PMERR_INVALID_DATATYPE	An invalid data type was specified.
PMERR_INVALID_DST_CODEPAGE	The destination code page parameter is invalid.
PMERR_INVALID_ERRORINFO_HANDLE	On WinFreeErrorInfo, the ERRINFO is not the handle of an ERRINFO structure, that is, it was not created by WinGetErrorInfo.
PMERR_INVALID_FLAG	An invalid bit was set for a parameter. Use constants defined by PM for options, and do not set any reserved bits.
PMERR_INVALID_FREE_MESSAGE_ID	An invalid message identifier was specified. The call has completed by assuming the message parameter and reply data types to be ULONG.
PMERR_INVALID_GROUP_HANDLE	An invalid program-group handle was specified.
PMERR_INVALID_HACCEL	An invalid accelerator-table handle was specified.
PMERR_INVALID_HAPP	The application handle passed to WinTerminateApp does not correspond to a valid session.
PMERR_INVALID_HATOMTBL	An invalid atom-table handle was specified.

PMERR_INVALID_HEAP_POINTER	An invalid pointer was found within the heap.
PMERR_INVALID_HEAP_SIZE	Invalid data was found within the heap.
PMERR_INVALID_HEAP_SIZE_PARM	Invalid data was found within the heap.
PMERR_INVALID_HEAP_SIZE_WORD	Invalid data was found within the heap.
PMERR_INVALID_HENUM	An invalid enumeration handle was specified.
PMERR_INVALID_HHEAP	An invalid heap handle was specified.
PMERR_INVALID_HMQ	An invalid message-queue handle was specified.
PMERR_INVALID_HPTR	An invalid pointer handle was specified.
PMERR_INVALID_HSTRUCT	An invalid (null) structure handle was specified.
PMERR_INVALID_HWND	An invalid window handle was specified.
PMERR_INVALID_INI_FILE_HANDLE	An invalid initialization-file handle was specified.
PMERR_INVALID_INTEGER	The specified atom is not a valid integer atom.
PMERR_INVALID_INTEGER_ATOM	The specified atom is not a valid integer atom.
PMERR_INVALID_MESSAGE_ID	A message identifier is invalid.
PMERR_INVALID_NUMBER_OF_PARMS	The number of parameters is invalid.
PMERR_INVALID_NUMBER_OF_TYPES	The function call has an invalid number (zero) of types.
PMERR_INVALID_PARAMETER	An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range -32,768 to +32,767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.
PMERR_INVALID_PARAMETER_TYPE	A parameter type is invalid for a bundle mask.
PMERR_INVALID_PARAMETERS	An application parameter value is invalid for its converted PM type. For example: a 4-byte value outside the range -32 768 to +32 767 cannot be converted to a SHORT, and a negative number cannot be converted to a ULONG or USHORT.

PMERR_INVALID_PARAMETERS	
PMERR_INVALID_PARM	A parameter to the function contained invalid data.
PMERR_INVALID_PROGRAM_HANDLE	An invalid program handle was specified.
PMERR_INVALID_SESSION_ID	The specified session identifier is invalid. Either zero (for the application's own session) or a valid identifier must be specified.
PMERR_INVALID_SRC_CODEPAGE	The source code page parameter is invalid.
PMERR_INVALID_STRING_PARM	The specified string parameter is invalid.
PMERR_INVALID_SWITCH_HANDLE	An invalid Window List entry handle was specified.
PMERR_INVALID_TARGET_HANDLE	An invalid target program-group handle was specified.
PMERR_INVALID_TITLE	The specified program or group title is too long or contains invalid characters.
PMERR_INVALID_TYPE_FOR_LENGTH	The data type for a control length is invalid.
PMERR_INVALID_TYPE_FOR_MPARAM	The message parameter type for a control MPARAM is invalid, that is, not mparam1, mparam2 or mreply.
PMERR_INVALID_TYPE_FOR_OFFSET	The data type for a control offset is invalid.
PMERR_INVALID_WINDOW	The window specified with a Window List call is not a valid frame window.
PMERR_KERNING_NOT_SUPPORTED	Kerning was requested on GpiCreateLogFont call to a presentation space associated with a device context that does not support kerning.
PMERR_LABEL_NOT_FOUND	The specified element label did not exist.
PMERR_MATRIX_OVERFLOW	An internal overflow error occurred during matrix multiplication. This can occur if coordinates or matrix transformation elements (or both) are invalid or too large.
PMERR_MEMORY_ALLOC	An error occurred during memory management.
PMERR_MEMORY_ALLOCATION_ERR	An error occurred during memory management.
PMERR_MEMORY_DEALLOCATION_ERR	An error occurred during memory management.
PMERR_METAFILE_IN_USE	An attempt has been made to

	access a metafile that is in use by another thread.
PMERR_METAFILE_INTERNAL_ERROR	An internal inconsistency has been detected during metafile unlock processing.
PMERR_METAFILE_LIMIT_EXCEEDED	The maximum permitted metafile size limit was exceeded during metafile recording.
PMERR_MSG_QUEUE_ALREADY_EXISTS	An attempt to create a message queue for a thread failed because a message queue already exists for the calling thread.
PMERR_MSGID_TOO_SMALL	The message identifier specified is too small.
PMERR_NEGATIVE_STRCOND_DIM	A negative array dimension was passed for a data type length.
PMERR_NO_BITMAP_SELECTED	An attempt has been made to operate on a memory device context that has no bit map selected.
PMERR_NO_CURRENT_ELEMENT	An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open element.
PMERR_NO_CURRENT_SEG	An attempt has been made to issue GpiQueryElementType or GpiQueryElement while there is no currently open segment.
PMERR_NO_FILL	No flood fill occurred because either the starting point color was the same as the input color when a boundary fill was requested, or the starting point color was not the same as the input color when a surface fill was requested.
PMERR_NO_METAFILE_RECORD_HANDLE	The metafile record handle was not found during metafile recording, or DevEscape (DEVESC_STARTDOC) was not issued when drawing to a OD_QUEUED device context with a *pszDataType* field of PM_Q_STD.
PMERR_NO_MSG_QUEUE	
PMERR_NO_PALETTE_SELECTED	An attempt to realize a palette failed because no palette was previously selected into the Presentation Space.
PMERR_NO_SPACE	The limit on the number of Window List entries has been reached with WinAddSwitchEntry.

PMERR_NOT_CREATED_BY_DEVOPENDC	An attempt has been made to destroy a device context using DevCloseDC that was not created using DevOpenDC.
PMERR_NOT_CURRENT_PL_VERSION	An unexpected data format was found in the initialization file.
PMERR_NOT_DRAGGING	A drag operation is not in progress at this time.
PMERR_NOT_IN_A_PM_SESSION	An attempt was made to access function that is only available from PM programs from a non-PM session.
PMERR_NOT_IN_AREA	An attempt was made to end an area using GpiEndArea or during segment drawing while not in an area bracket.
PMERR_NOT_IN_DRAW_MODE	An attempt was made to issue GpiSavePS or GpiRestorePS while the drawing mode was not set to DM_DRAW.
PMERR_NOT_IN_ELEMENT	An attempt was made to end an element using GpiEndElement or during segment drawing while not in an element bracket.
PMERR_NOT_IN_IDX	The application name, key-name or program handle was not found.
PMERR_NOT_IN_IMAGE	An attempt was made to end an image during segment drawing while not in an image bracket.
PMERR_NOT_IN_PATH	An attempt was made to end a path using GpiEndPath or during segment drawing while not in a path bracket.
PMERR_NOT_IN_RETAIN_MODE	An attempt was made to issue a segment editing element function that is invalid when the actual drawing mode is not set to **retain**.
PMERR_NOT_IN_SEG	An attempt was made to end a segment using GpiCloseSegment while not in a segment bracket.
PMERR_NOT_SELF_DESCRIBING_DTYP	A data type is not self-describing.
PMERR_OK	The code page or country code passed is not valid.
PMERR_OK	The window handle or menu item passed is not valid.
PMERR_OPENING_INI_FILE	Unable to open initialization file (due to lack of disk space

	for example).
PMERR_ORDER_TOO_BIG	An internal size limit was exceeded while converting orders from short to long format during GpiPutData processing. An order was too long to convert.
PMERR_OWN_SET_ID_REFS	An attempt to unload a font failed because the setid is still being referenced.
PMERR_PALETTE_BUSY	An attempt has been made to reset the owner of a palette when it was busy.
PMERR_PALETTE_SELECTED	Color palette operations cannot be performed on a presentation space while a palette is selected.
PMERR_PARAMETER_OUT_OF_RANGE	The value of a parameter was not within the defined valid range for that parameter.
PMERR_PATH_INCOMPLETE	An attempt was made to open or close a segment either directly or during segment drawing, or to issue GpiAssociate while there is an open path bracket.
PMERR_PATH_LIMIT_EXCEEDED	An internal size limit was exceeded during path or area processing.
PMERR_PATH_UNKNOWN	An attempt was made to perform a path function on a path that did not exist.
PMERR_PEL_IS_CLIPPED	An attempt was made to query a pel that had been clipped using GpiQueryPel.
PMERR_PEL_NOT_AVAILABLE	An attempt was made to query a pel that did not exist in GpiQueryPel (for example, a memory device context with no selected bit map).
PMERR_PRINTER_DD_NOT_DEFINED	The Presentation Manager device driver has not been defined.
PMERR_PRINTER_QUEUE_NOT_DEFINED	The spooler queue for the printer has not been defined.
PMERR_PRN_ADDR_IN_USE	A printer is already defined on the port.
PMERR_PRN_ADDR_NOT_DEFINED	The printer port has not been defined.
PMERR_PRN_NAME_NOT_DEFINED	The printer has not been defined.
PMERR_PROLOG_ERROR	A prolog error was detected during drawing. Segment prologs are used internally within retained segments and also appear in metafiles. This

	error can also arise from an End Prolog order that is outside a prolog.
PMERR_PS_BUSY	An attempt was made to access the presentation space from more than one thread simultaneously.
PMERR_PS_IS_ASSOCIATED	An attempt was made to destroy a presentation or associate a presentation space that is still associated with a device context.
PMERR_PS_NOT_ASSOCIATED	An attempt was made to access a presentation space that is not associated with a device context.
PMERR_QUEUE_TOO_LARGE	An attempt to create a message queue has failed because the value specified for the size of the message queue is too large.
PMERR_RASTER_FONT	A request was made for the outline of a bit-map font. Outlines can only be returned for vector font characters.
PMERR_REALIZE_NOT_SUPPORTED	An attempt was made to create a realizable logical color table on a device driver that does not support this function.
PMERR_REGION_IS_CLIP_REGION	An attempt was made to perform a region operation on a region that is selected as a clip region.
PMERR_RESOURCE_DEPLETION	An internal resource depletion error has occurred.
PMERR_RESOURCE_NOT_FOUND	The specified resource identity could not be found.
PMERR_SEG_AND_REFSEG_ARE_SAME	The segid and refsegid specified with GpiSetSegmentPriority were the same.
PMERR_SEG_CALL_STACK_EMPTY	A call stack empty condition was detected when attempting a pop function during GpiPop or segment drawing.
PMERR_SEG_CALL_STACK_FULL	A call stack full condition was detected when attempting to call a segment using GpiCallSegmentMatrix, attempting to preserve an attribute, or during segment drawing.
PMERR_SEG_IS_CURRENT	An attempt was made to issue GpiGetData to a segment that was currently open.
PMERR_SEG_NOT_CHAINED	An attempt was made to

	issue GpiDrawFrom, GpiCorrelateFrom or GpiQuerySegmentPriority for a segment that was not chained.
PMERR_SEG_NOT_FOUND	The specified segment identifier did not exist.
PMERR_SEG_OVFLOW	The input PIF has more than 1000 called segments. This has overflowed an internal buffer.
PMERR_SEG_STORE_LIMIT_EXCEEDED	The maximum permitted retained segment store size limit was exceeded.
PMERR_SETID_IN_USE	An attempt was made to specify a setid that was already in use as the currently selected character, marker or pattern set.
PMERR_SETID_NOT_FOUND	An attempt was made to delete a setid that did not exist.
PMERR_SMB_OVFLOW	The input PIF has more than 100 symbol sets defined. This has overflowed an internal buffer.
PMERR_SOMDD_IS_ACTIVE	The DSOM daemon is already active.
PMERR_SOMDD_NOT_STARTED	The DSOM daemon failed to start.
PMERR_SOURCE_SAME_AS_TARGET	The direct manipulation source and target process are the same.
PMERR_SPL_CANNOT_OPEN_FILE	Unable to open the file.
PMERR_SPL_DD_NOT_FOUND	The Presentation Manager device driver definition could not be found.
PMERR_SPL_DEVICE_ALREADY_EXISTS	The device already exists.
PMERR_SPL_DEVICE_LIMIT_REACHED	The limit on the number of devices has been reached.
PMERR_SPL_DEVICE_NOT_INSTALLED	The device has not been installed.
PMERR_SPL_DRIVER_ERROR	No Presentation Manager device driver supplied or found.
PMERR_SPL_DRIVER_NOT_INSTALLED	The Presentation Manager device driver has not been installed.
PMERR_SPL_FILE_NOT_FOUND	Unable to find the file.
PMERR_SPL_HARD_NETWORK_ERROR	Hard network error.
PMERR_SPL_INI_FILE_ERROR	Error accessing the initialization file.
PMERR_SPL_INV_DATATYPE	The spool file data type is

	invalid.
PMERR_SPL_INV_DRIVER_DATATYPE	The data type is invalid for the Presentation Manager device driver.
PMERR_SPL_INV_FORMS_CODE	The forms code for the job is invalid.
PMERR_SPL_INV_HSPL	The spooler handle is invalid.
PMERR_SPL_INV_JOB_ID	The job id is invalid.
PMERR_SPL_INV_LENGTH_OR_COUNT	The length or count is invalid.
PMERR_SPL_INV_PRIORITY	The priority for the job is invalid.
PMERR_SPL_INV_PROCESSOR_DATTYPE	The data type is invalid for the spooler queue processor.
PMERR_SPL_INV_QUEUE_NAME	The spooler queue name is invalid.
PMERR_SPL_INV_TOKEN	The token is invalid.
PMERR_SPL_JOB_NOT_PRINTING	The print job is not printing.
PMERR_SPL_JOB_PRINTING	The print job is already printing.
PMERR_SPL_MANY_QUEUES_ASSOC	More than one queue has been associated with the printer.
PMERR_SPL_NO_CURRENT_FORMS_CODE	There is no current forms code defined to the Presentation Manager device driver.
PMERR_SPL_NO_DATA	No data supplied or found.
PMERR_SPL_NO_DEFAULT_QUEUE	There is no default spooler queue for the printer.
PMERR_SPL_NO_DISK_SPACE	There is not enough free disk space.
PMERR_SPL_NO_FREE_JOB_ID	There is no free job id available.
PMERR_SPL_NO_MEMORY	There is not enough free memory.
PMERR_SPL_NO_QUEUES_ASSOCIATED	A queue has not been associated with the printer.
PMERR_SPL_NO_SUCH_LOG_ADDRESS	The logical address does not exist (that is, it is not defined in the initialization file).
PMERR_SPL_NOT_AUTHORISED	Not authorized to perform the operation.
PMERR_SPL_PRINT_ABORT	The job has already been aborted.
PMERR_SPL_PRINTER_NOT_FOUND	The printer definition could not be found.
PMERR_SPL_PROCESSOR_ERROR	No spooler queue processor supplied or found.

PMERR_SPL_PROCESSOR_NOT_INST	The spooler queue processor has not been installed.
PMERR_SPL_QUEUE_ALREADY_EXISTS	The spooler queue already exists.
PMERR_SPL_QUEUE_ERROR	No spooler queue supplied or found.
PMERR_SPL_QUEUE_NOT_EMPTY	The spooler queue contains print jobs.
PMERR_SPL_QUEUE_NOT_FOUND	The spooler queue definition could not be found.
PMERR_SPL_SPOOLER_NOT_INSTALLED	The spooler is not installed.
PMERR_SPL_STATUS_STRING_TRUNC	The print job status string has been truncated.
PMERR_SPL_TEMP_NETWORK_ERROR	Temporary network error.
PMERR_SPL_TOO_MANY_OPEN_FILES	Too many open files.
PMERR_SPOOLER_QP_NOT_DEFINED	The spooler queue processor has not been defined.
PMERR_START_POINT_CLIPPED	The starting point specified for flood fill is outside the current clipping path or region.
PMERR_STARTDOC_NOT_ISSUED	A request to write spooled output without first issuing a STARTDOC was attempted.
PMERR_STARTED_IN_BACKGROUND	The application started a new session in the background.
PMERR_STOP_DRAW_OCCURRED	Segment drawing or GpiPlayMetaFile was stopped prematurely in response to a GpiSetStopDraw request.
PMERR_TOO_MANY_METAFILES_IN_USE	The maximum number of metafiles allowed for a given process was exceeded.
PMERR_TRUNCATED_ORDER	An incomplete order was detected during segment processing.
PMERR_UNABLE_TO_CLOSE_DEVICE	Unable to close the print device (for example, powered off or offline).
PMERR_UNCHAINED_SEG_ZERO_INV	An attempt was made to open segment with segment identifier zero and the ATTR_CHAINED segment attribute not specified.
PMERR_UNSUPPORTED_ATTR	An unsupported attribute was specified in the attrmask with GpiSetAttrs or GpiQueryAttrs.
PMERR_UNSUPPORTED_ATTR_VALUE	An attribute value was specified with GpiSetAttrs that is not supported.
PMERR_WINDOW_LOCK_OVERFLOW	An overflow occurred for the

	use count of a window.
PMERR_WINDOW_LOCK_UNDERFLOW	An attempt was made to decrement the use count of a window below zero.
PMERR_WINDOW_NOT_LOCKED	The window specified in WinSendMsg was not locked.
PMERR_WPDSERVER_IS_ACTIVE	The Workplace Shell DSOM Server is already active.
PMERR_WPDSERVER_NOT_STARTED	The Workplace Shell DSOM Server could not be started.
WPERR_INVALID_FLAGS	An invalid flag was specified.
WPERR_INVALID_OBJECTID	An invalid object ID was specified.
WPERR_INVALID_TARGET_OBJECT	An invalid target object was specified.

Area and Polygon Primitives

An *area* is one or more closed figures that can be drawn filled, outlined, or filled and outlined. If an area includes more than one figure, those figures can be separate or intersecting.

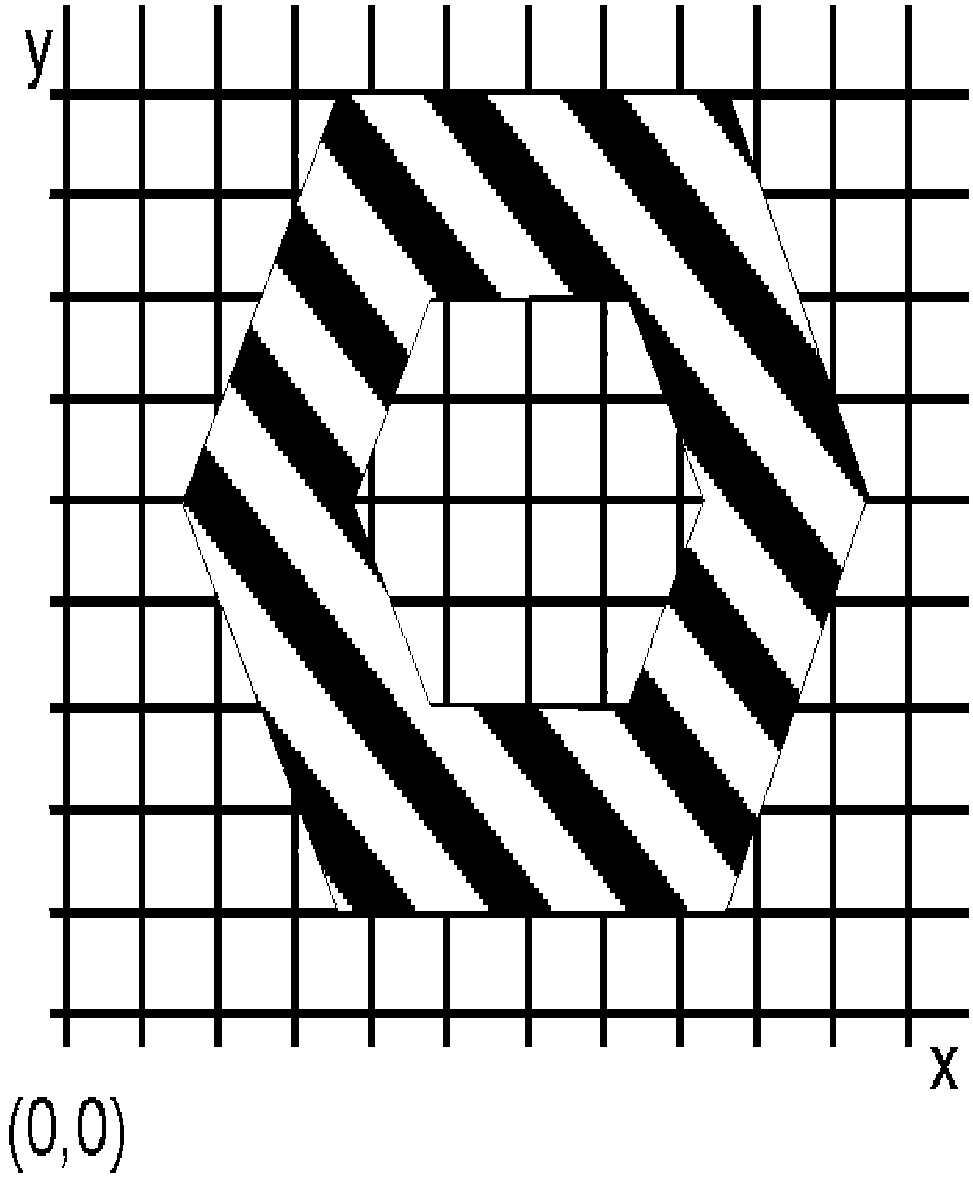
A *polygon*, too, is one or more closed figures that can be drawn filled, outlined, or filled and outlined. Polygons, unlike areas, are limited to figures with straight edges. Polygons also can be separate or intersecting.

The following topics are related to information in this chapter:

- Presentation spaces
- Line and arc primitives
- Color and mix attributes
- Fonts
- Bit maps
- Paths

About Area Primitives

Applications can create, outline, and fill areas and can create custom-fill patterns from bit maps or font symbols. Some graphics functions are not valid within an area definition. For example, you cannot include marker, image, or character-string primitives in an area definition. The following figure is an example of an area.



An Area

This area comprises two hexagons, one completely enclosed by the other. The area is filled with the current area-fill pattern.

Attributes of Area Primitives

The attributes of the area primitive are contained in a data structure called AREABUNDLE. These attributes are:

- Pattern symbol
- Pattern reference point
- Pattern set
- Foreground color
- Background color
- Foreground mix attribute
- Background mix mode

When an application creates a presentation space, the area attributes are set to the default values shown in the following table.

Area Attribute Default Values

Attribute	Default Value	Function that Redefines Attribute
Pattern symbol	solid	GpiSetPattern
Pattern reference point	(0,0)	GpiSetPatternRefPoint
Pattern set	LCID_DEFAULT	GpiSetPatternSet
Foreground color	Black	GpiSetAttrs (ABB_COLOR)
Background color	Clear	GpiSetAttrs (ABB_BACK_COLOR)
Foreground mix	Overpaint	GpiSetAttrs (ABB_MIX_MODE)
Background mix	Leave alone	GpiSetAttrs (ABB_BACK_MIX_MODE)

Note: If the default (LCID_DEFAULT) for pattern set is changed, the base pattern set cannot be reselected with GpiSetPatternSet.

Pattern Symbol Attribute

The *current pattern symbol*, also called the *symbol point code*, is selected from the current pattern set with GpiSetPattern. The pattern symbol selected for the specified presentation space is used as the subsequent fill pattern until a new symbol is selected. Applications must not call GpiSetPattern while in an area or a path. If the current pattern set specifies a bit map, this attribute is ignored.

The following table describes the pattern symbols provided by the PM programming interface in a base pattern set. These symbols are not necessarily available from other pattern sets.

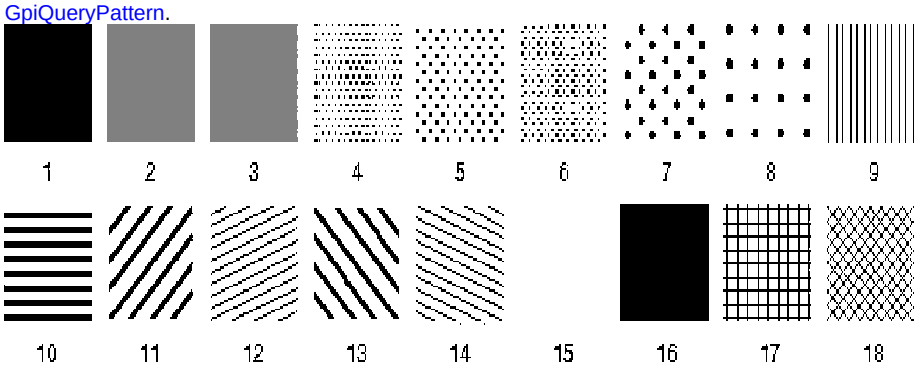
The Base Pattern Set

Symbol	Identifier	Long Value
Solid shading decreasing in dots per inch	PATSYM_DENSE1 through PATSYM_DENSE8	1L - 8L
Vertical lines	PATSYM_VERT	9L
Horizontal lines	PATSYM_HORIZ	10L
Lines bottom left to top right	PATSYM_DIAG1	11L
Lines bottom left to middle right	PATSYM_DIAG2	12L
Lines top left to bottom right	PATSYM_DIAG3	13L
Lines top left to middle right	PATSYM_DIAG4	14L
No shading	PATSYM_NOSHADE	15L
Solid shading	PATSYM_SOLID	16L
Alternate pels	PATSYM_HALFTONE	17L
Cartesian grid	PATSYM_HATCH	18L
Diagonal crosshatch	PATSYM_DIAGHATCH	19L
Blank (often called the *clear* pattern)	PATSYM_BLANK	64L

The default pattern symbol (PATSYM_DEFAULT) is identical to the PATSYM_SOLID symbol, and has a long value of 0L. The error pattern

symbol (PATSYM_ERROR) has a long value of -1L.

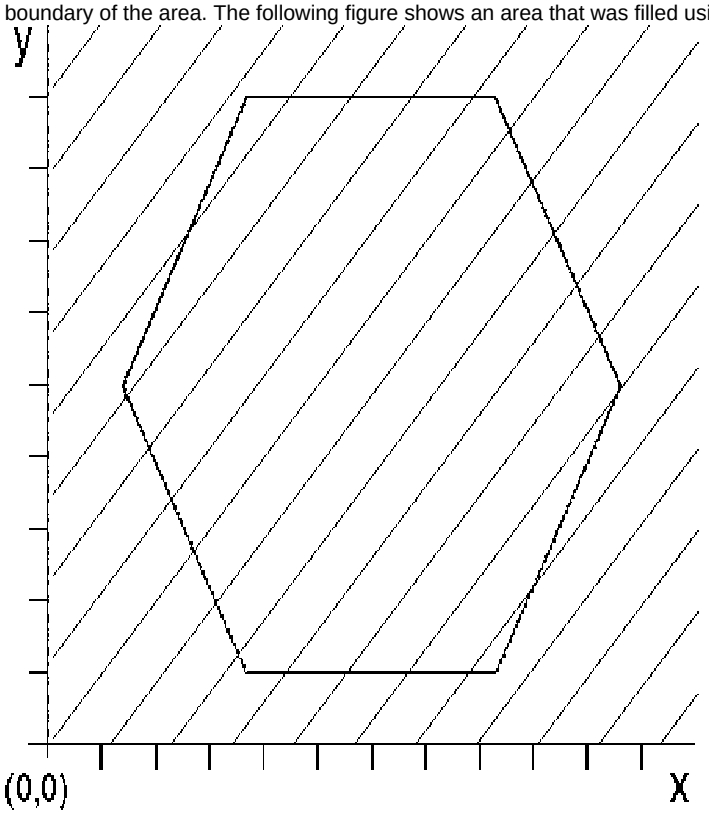
The following figure shows the patterns from the base pattern symbol set. Applications can determine the current pattern set by calling GpiQueryPattern.



The Base Pattern Symbol Set

Pattern Reference Point Attribute

The *pattern reference point* is the point from which the area's fill-pattern spreads horizontally and vertically. The lower-left corner of the pattern is aligned on this point. The pattern reference point has a default value of (0,0), and is expressed in world coordinates. Applications can determine the pattern reference point with GpiQueryPatternRefPoint. Applications can specify the pattern reference point with GpiSetPatternRefPoint; however, this should not be done inside an area. The pattern reference point does not have to be within the boundary of the area. The following figure shows an area that was filled using the default pattern reference point.



The Pattern Reference Point

The pattern of diagonal lines is aligned with the pattern reference point (0,0). When the picture is displayed or printed, only the hexagon will be filled.

Use of the default reference point causes the diagonal lines of the pattern to intersect the area boundary at specific points. If you change the reference point to (0,4), for example, the pattern shifts upward, and the points of intersection with the area boundary are different. Although the reference point is outside the area, when an application displays or prints the picture, only the area is filled.

Pattern Set Attribute

When you create a presentation space, the current pattern set and other area attributes are set to the default. The current pattern-a black solid, as described earlier- is specified from the supplied pattern set. (The supplied set of pattern symbols contains image or raster patterns only.) An area primitive is filled by repeating this pattern vertically and horizontally within the area boundary.

Default Pattern Set

The standard pattern set contains patterns of solid shading with decreasing intensity, and patterns of vertical, horizontal, and diagonal lines. If the default pattern set is changed with GpiSetDefAttrs, its patterns are not accessible. The patterns from the default set can be recovered by specifying GpiSetDefAttrs with the value LCID_DEFAULT, (0).

Customizing Pattern Sets

An application can create custom patterns by using bit maps or characters and symbols from an image font. When designing custom patterns, consider that the operating system uses only the first eight bits in the first eight rows, starting with the lower-left corner of the bit map. The cell size of an image font used as a fill pattern might be too large to include the complete character or symbol.

An application can change the appearance of a fill pattern by changing its alignment in an area primitive with GpiSetPatternRefPoint. When the default reference point is set, the operating system aligns the lower-left corner of the fill pattern with the point (0,0) in the application's world space and begins filling the area. If an application adjusts the pattern reference point to (3,2), The operating system aligns the lower-left corner of the fill pattern with the point (3,2) in the application's world space and begins filling the area. By moving the reference point from (0,0) to (3,2), the fill pattern appears shifted up two pels and to the right three pels.

Custom Fill Patterns from a Bit Map

To create a custom pattern from a hard-coded bit map, an application must use the following steps:

1. Create or load a bit map to obtain a bit-map handle; for example, use GpiLoadBitmap.
2. Call GpiSetBitmapId (ID) to tag the bit map with a local identifier (lcid) from 1 to 254.
3. Call GpiSetPatternSet (ID) or GpiSetAttrs (ABB_SET) to set the current fill pattern.

The application can now draw the area.

Custom Fill Patterns from a Font Symbol

To create a custom pattern from a character or a symbol in a font:

1. Create a logical image font and assign it an ID; for example use GpiCreateLogFont.
2. Call GpiSetPatternSet (ID) or GpiSetAttrs (ABB_SET), passing it the local identifier for the font.
3. Call GpiSetPattern, passing the value of the code point for a character or symbol in the font.

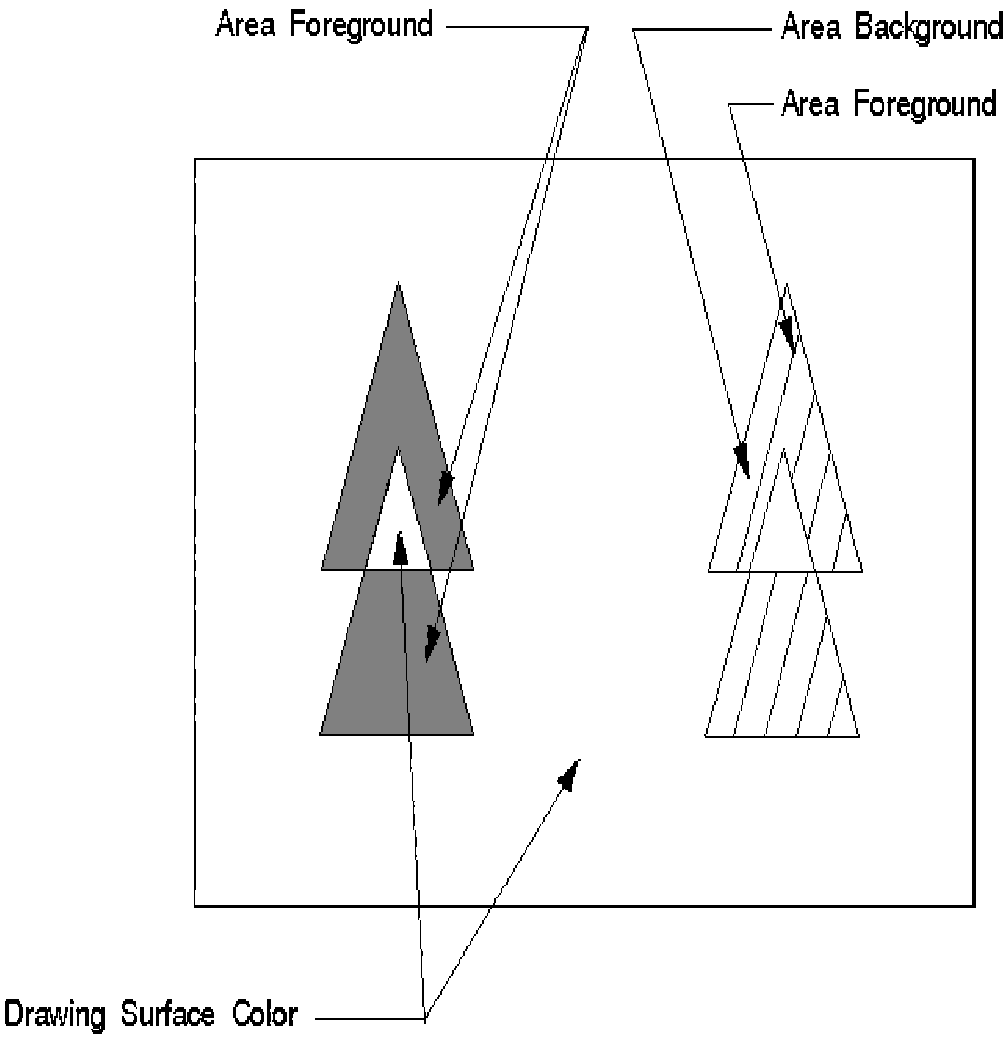
The application can now draw the area.

Area Colors and Mix Attributes

The color attribute defines the color used to draw a primitive or an object. The mix attribute determines how the color of a primitive or an object is combined with the color of the drawing surface, or any other objects on the surface. Both attributes also are described in Color and Mix Attributes.

The area color defines the color used to fill the output from any of the IBM OS/2 area functions, when necessary. The area primitive is the only primitive in which the color can be changed during the drawing of the area. Therefore, this attribute, more than the other area attributes, depends on the *current value* as described in Attribute Currentness.

When a presentation space is created, the area color initial default is black. The pattern symbol initial default, explained previously, is solid. Areas are one of the primitives that have a foreground and background color, as shown in the following figure. The appearance of the area also depends on the pattern symbol.



Area Primitives

Area primitives have both a color and background color attribute. The background color does not appear if the pattern symbol is solid, and the background color is undefined if the pattern symbol is a customized, multi-colored bit map.

When a presentation space is created, the area mix attribute initial default is FM_OVERPAINT. The *overpaint* mix attribute specifies that the area color is not to be modified by the color of the drawing surface. If the area mix attribute is changed, the area color is mixed with colors that are already on the drawing surface.

When the pattern symbol is solid, the color that fills the area, if necessary, is the current area foreground color. If the pattern symbol is changed to be a pattern of vertical lines, for example, the color of those lines is the current area foreground color. Inside of the area bracket, there can be other colors if these colors are the current color for primitives such as lines or arcs, that are drawn within the area bracket. These colors are not specifically defined or influenced by the area foreground color.

The area background color initial default is CLR_BACKGROUND. Usually this is defined by the application to the same color as the drawing surface. If the pattern symbol is solid, the area background color *does not* appear. If the pattern symbol is changed to be a pattern of vertical lines, for example, the color in between those lines is the current area background color.

The area background mix attribute initial default is BM_LEAVEALONE. The *leave-alone* background mix attribute specifies that the area background color is not drawn. This means that for a nonsolid pattern symbol, the drawing-surface color or the color of an object, not created by the area bracket, but on the drawing surface, shows through the nonsolid pattern. The area background color, for nonsolid pattern symbols, appears only if the background mix attribute is changed to overpaint, BM_OVERPAINT.

If you have customized the pattern symbol with a bit map, the color definitions of the area primitive change. The foreground color corresponds to the color of the pels that are specified ON in the fill-pattern bit map. The background color corresponds to the color of the pels that are specified OFF in the fill-pattern bit map. The foreground and background mix attribute do not change.

Note: Background color and mix attribute only apply to monochrome (2-color) fill patterns. The bit set to 0 is defined as the background and the bit set to 1 is defined as the foreground.

A multi-colored bit map is unaffected by the background color and mix attributes.

To specify a new color or mix attribute call GpiSetAttrs. This function accepts as input the type of primitive, for example PRIM_AREA, a list of attributes that are to be changed, a list of attributes that are to be set to their default values, and the values for the attributes that are to be changed. GpiSetAttrs is useful to specify colors and mix attributes just for a specific data structure -for example the AREABUNDLE structure. GpiSetAttrs also provides some protection against invalid colors.

To determine the current area color and mix attribute, call GpiQueryAttrs. This function accepts as input the primitive type and the attributes in question. It returns as output an array of values for the specifically queried attributes.

To reset the default area color and mix attribute, just as with any other attribute specified in the AREABUNDLE. data structure, call GpiSetDefAttrs. This function accepts as input the type of primitive, for example PRIM_AREA, the attributes to be changed, and the values that will become the new default values. The changing of default values is important when working with segments. Changing the default values during a series of drawing functions is not recommended.

The area color and mix attribute also can be specified with:

- GpiSetColor
- GpiSetMix
- GpiSetBackColor
- GpiSetBackMix

However, these functions have the disadvantage of specifying the foreground and background color or mix attribute for all primitive BUNDLE data structures that have the respective component.

There are four queries that determine the color and mix attribute as specified by GpiSet... functions:

- GpiQueryColor
- GpiQueryMix
- GpiQueryBackColor
- GpiQueryBackMix

If the area color, area background color, mix attribute, or background mix attribute were specified individually, the aforementioned queries can return a value inconsistent with the current area attribute.

Area Brackets

Areas also are referred to as *area brackets* , because the functions that create and define the area always are "bracketed" by the two functions, GpiBeginArea and GpiEndArea. Only one area is defined between these two functions. However, within the area, there can be any number of adjacent, intersecting, or completely separate area primitives.

GpiBeginArea signals the start of a group of primitives that define the boundary of the area. The current position is not changed by GpiBeginArea, although it is updated by any drawing instructions that follow.

If you call GpiSetCurrentPosition or GpiMove within an area definition, the current figure is closed automatically, and a new figure is started at the current position by the next line or curve. A figure in the area, whose start and end points are not the same, is closed by drawing a straight line from the current position to the start position of the figure.

GpiEndArea signals the end of an area definition and tells the operating system to draw the area. If an application does not close a figure before calling GpiEndArea, the figure is closed automatically by drawing a straight line from the end point of the last line or curve to the start point of the current figure. The current position is always updated to the end point of the last line drawn.

The functional sequence to draw the area in the following figure could be:

```
#include <os2.h>
void fncAREA01(void){

#if 0

GpiBeginArea                /* Starts the area bracket          */

GpiSetCurrentPosition       /* Set the start point of the       */
                            /* outer hexagon.                   */
GpiPolyLine                 /* Draw the outer hexagon.          */
GpiSetCurrentPosition       /* Set the start point of the inner */
                            /*  hexagon.                        */
GpiPolyLine                 /* Draw the inner hexagon.          */

GpiEndArea                  /* End the area definition.         */

#endif

}
```

An area bracket contains the functions that define an area. Only one area can be defined between "bracket" functions.

Following the beginning of an area bracket, an application can define the shape and location of the area in world space by using the following line and arc functions:

- GpiBox
- GpiFullArc
- GpiLine
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolyLine
- GpiPolySpline

If an application calls either GpiBox or GpiFullArc inside an area, it must use the DRO_OUTLINE option.

In addition to the drawing functions, an application also can use the following specification functions inside an area bracket:

- GpiBeginElement
- GpiCallSegmentMatrix
- GpiComment
- GpiElement
- GpiEndArea
- GpiEndElement
- GpiGetData
- GpiLabel
- GpiMove
- GpiOffsetElementPointer
- GpiPop
- GpiPutData
- GpiSetArcParams
- GpiSetAttrMode

- GpiSetAttrs
- GpiSetColor
- GpiSetCurrentPosition
- GpiSetEditMode
- GpiSetElementPointer
- GpiSetElementPointerAtLabel
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMix
- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix

An application also can use the following query functions inside an area bracket:

- GpiQueryArcParams
- GpiQueryAttrMode
- GpiQueryAttrs
- GpiQueryBackColor
- GpiQueryBackMix
- GpiQueryBoundaryData
- GpiQueryCharAngle
- GpiQueryCharBox
- GpiQueryCharDirection
- GpiQueryCharMode
- GpiQueryCharSet
- GpiQueryCharShear
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt
- GpiQueryClipBox
- GpiQueryClipRegion
- GpiQueryColor
- GpiQueryColorData
- GpiQueryColorIndex
- GpiQueryCp
- GpiQueryCurrentPosition
- GpiQueryDefaultViewMatrix
- GpiQueryDefCharBox
- GpiQueryDevice
- GpiQueryDeviceBitmapFormats
- GpiQueryEditMode
- GpiQueryFontFileDescriptions
- GpiQueryFontMetrics
- GpiQueryFonts
- GpiQueryGraphicsField
- GpiQueryInitialSegmentAttrs
- GpiQueryKerningPairs
- GpiQueryLineEnd
- GpiQueryLineJoin
- GpiQueryLineType
- GpiQueryLineWidth
- GpiQueryLineWidthGeom
- GpiQueryLogColorTable
- GpiQueryMarker
- GpiQueryMarkerBox
- GpiQueryMarkerSet
- GpiQueryMix
- GpiQueryModelTransformMatrix
- GpiQueryNearestColor
- GpiQueryNumberSetIds
- GpiQueryPageViewport
- GpiQueryPattern
- GpiQueryPatternRefPoint
- GpiQueryPatternSet
- GpiQueryPel
- GpiQueryPickAperturePosition
- GpiQueryPickApertureSize
- GpiQueryRealColors
- GpiQueryRegionBox
- GpiQueryRegionRects
- GpiQueryRGBColor
- GpiQuerySegmentAttrs

- GpiQuerySegmentNames
- GpiQuerySegmentPriority
- GpiQuerySegmentTransformMatrix
- GpiQuerySetIds
- GpiQueryStopDraw
- GpiQueryTag
- GpiQueryViewingLimits
- GpiQueryViewingTransformMatrix
- GpiQueryWidthTable

Area Bracket Attributes

In addition to the attributes controlled by the AREABUNDLE data structure, there are two attributes that can be specified for each individual area bracket:

- Area boundary
- Area construction

These bracket attributes affect how the area primitives inside the bracket are drawn.

The concept of *attribute currentness* is especially important to areas and paths that contain multiple figures.

Area Boundaries

An application specifies the *area boundary* when it calls GpiBeginArea. There are two options:

- BA_BOUNDARY (the default)
- BA_NOBOUNDARY.

The BA_BOUNDARY value tells the programming interface to draw all outlines of the area primitives within the area bracket, using a line that conforms to the current LINEBUNDLE attributes. If the line attributes have not been changed, the default outline color is black on most displays and printers, and the default line style is solid. An application can change the line color and line styles within the area bracket. The interior of the area primitive is filled with the current pattern.

To prevent the interface from outlining an area, an application can use the BA_NOBOUNDARY flag when calling GpiBeginArea. Only the interior fill pattern is visible. This value most often is used when an area contains many overlapping figures, and the interior lines are not desired.

You could think of this option in terms similar to the outline and fill options discussed with boxes and full arcs. The BA_BOUNDARY value corresponds to DRO_OUTLINEFILL, and BA_NOBOUNDARY, to DRO_FILL. There is no distinct boundary value that corresponds to DRO_OUTLINE, the simple outline. To simulate this effect, OR the BA_NOBOUNDARY value with the appropriate area construction value when calling GpiBeginArea.

Area Construction

An application specifies the *area construction* when it calls GpiBeginArea. There are two options:

- BA_ALTERNATE (the default)
- BA_WINDING.

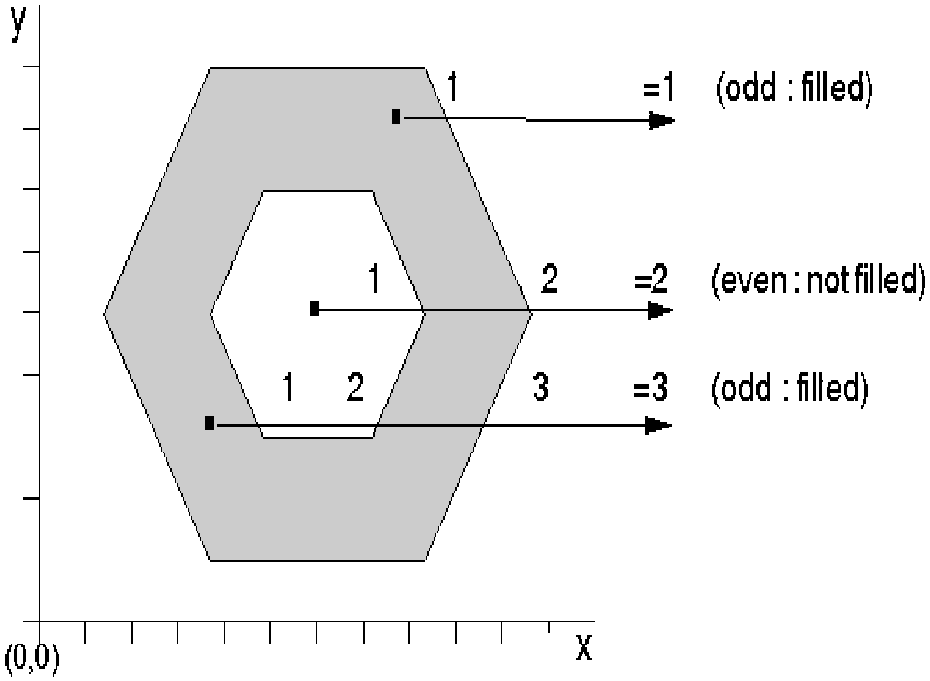
Construction modes also are called *area-fill* modes. They provide different ways of determining whether a given point is included in the filled area. Normally, the construction mode you use is a matter of personal preference.

Alternate Mode

In alternate mode, the following occurs:

- A point is included in the filled area if you have to cross an odd number of lines in the area when drawing a line from that point to infinity.
- A point is not included in the filled area if you have to cross an even number of lines in the area when drawing a line from that point to infinity.

In the example in the following figure, the inner hexagon is not shaded, because to draw a line from any point in the hexagon to infinity, you must cross two boundary lines. The remainder of the area is shaded, because to move outside the area, you need to cross only one boundary line.



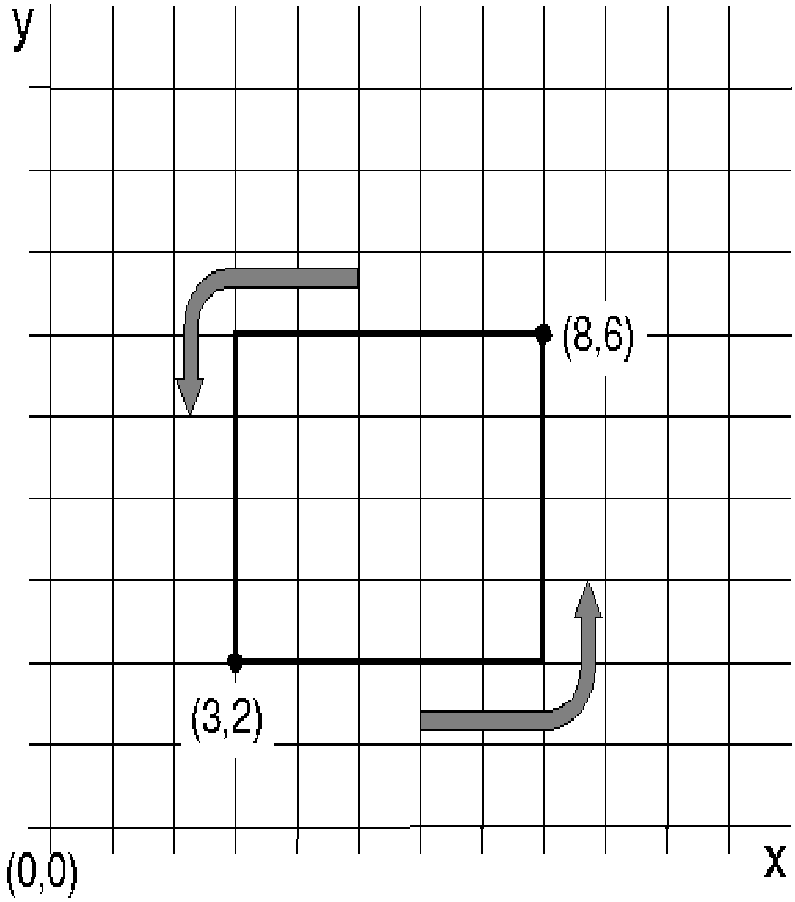
Calculating Filled Areas Constructed in Alternate Mode

Winding Mode

In winding mode, the direction in which the boundary lines in the area are drawn determines whether a given point is included in the filled area. The direction of a line depends on both the graphics functions used to draw it and the world coordinates that define it. For example, if the current position of a presentation space is (c,c), and GpiBox is called with the diagonally-opposite corner of the box specified as (d,d), GpiBox always draws the box in the following order:

1. (xc.,yc.) to
2. (xd.,yc.) to
3. (xd.,yd.) to
4. (xc.,yd.) and returning to
5. (xc.,yc.).

As illustrated in the following figure, in some cases the box is drawn in a counterclockwise direction. When either xd. is less than xc., or yd. is less than yc., but not both, a box is drawn clockwise.



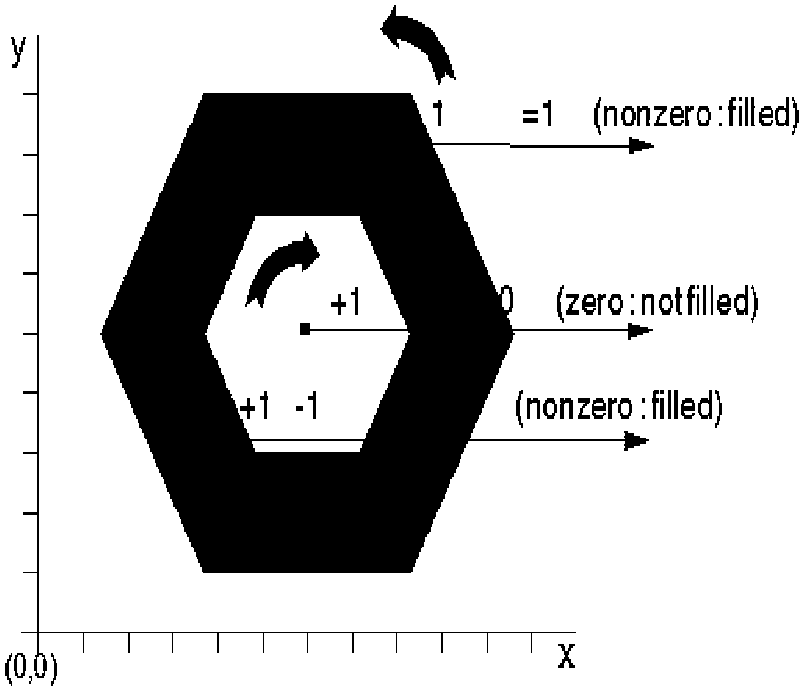
The Box

The current position is (3,2) and the specified corner is at (8,6). The box is drawn from (3,2) to (8,2) to (8,6) to (3,8) to (3,2).

For the polyline primitive, the direction in which a line is drawn depends on the relative values of the end points of each line, so you can choose whether to draw in a clockwise or counterclockwise direction.

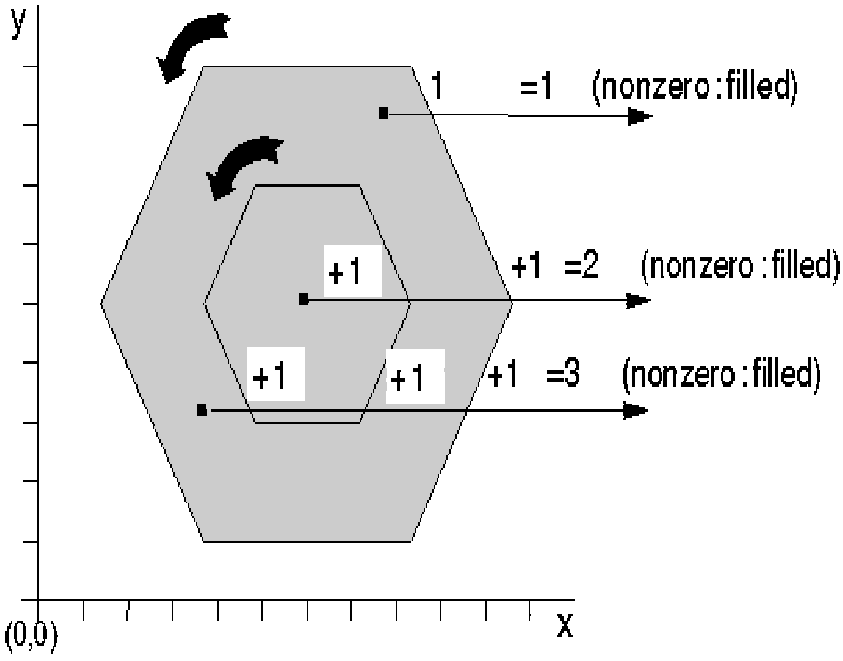
To determine if a given point is included in the filled area, count the number of lines to be crossed to move from that point to infinity. For each boundary line drawn in one direction, add one to the tally. For each line drawn in the opposite direction, subtract one from the tally. A point is within the area if the result is nonzero.

If two figures -for example, the hexagons in the following figure- are drawn in different directions, the tally for the inner hexagon is 0 and the area will look exactly as it does in alternate mode.



Area Constructed in Different Directions in Winding Mode

If the two hexagons are drawn in the same direction, the result is 2, and the inner hexagon is shaded as shown in the following figure.



Area Constructed in the Same Direction in Winding Mode

The boundary lines of the area in the previous figure have been drawn and are visible through the area-fill pattern. The boundary lines of an area primitive do not have to be drawn; but if they are, they are drawn according to the current line attributes.

To vary the appearance of different parts of the boundary line, you can change the current line attributes during area definition.

The boundary lines of an area are considered a part of the area's interior. Therefore, when you draw an area without boundary lines (BA_NOBOUNDARY), the area-fill pattern extends to include the boundaries of the area, and the area is the same size it would be if the boundary lines had been drawn.

Attribute Currentness

Graphics functions that change attributes are valid within the area bracket. There are misconceptions, however, about which attributes are used when the figures are displayed in a window or drawn to the printer.

Review the following functional sequence:

Function	Effect
GpiSetColor	Sets color to red.
GpiBeginArea	Opens area bracket
GpiLine	Draws a straight line
GpiSetColor	Sets color to green.
GpiPolyLine	Draws two additional lines to form a triangle.
GpiEndArea	Ends the area bracket and displays a picture.

If you are displaying on a color monitor, the image appears as follows:

The triangle has one red line, drawn with GpiLine and two green lines, drawn with GpiPolyLine. The interior of the triangle, if drawn, is red. Red is the color that was current when the area was defined and, therefore, is the current color for the area interior. In terms of the interior area, green is only the color that was current when the function that initiates drawing was called.

If the application immediately opened a new area bracket, the interior of that bracket, if drawn, would be green. The color attribute for the second AREABUNDLE, is not "inherited" from the last AREABUNDLE drawn.

About Polygon Primitives

Area brackets have the flexibility to draw any combination of curved or straight lines and combine the results into a closed figure. An advantage that polygons sometimes have over area brackets, however, is that they require fewer time-consuming calculations to provide that kind of flexibility.

The operating system provides a function that enables you to draw multiple straight-line closed areas outside of an area bracket. Like bracket-generated areas, they can be adjacent, intersecting, or completely separate.

The function is GpiPolygons and the set of objects it draws are called *polygon primitives* . A polygon primitive is any set of polygons with specified vertices that can be filled or filled and outlined. This function accepts as input the desired number of polygons, the POLYGON data structure for each polygon, and a boundary and construction option similar to those of the GpiBeginArea function.

The purpose of GpiPolygons is to enable you to specify an area in such a way as to pass all of the area boundaries at once. This improves performance significantly because the operating system cannot process the accumulated primitives inside an area bracket until it reaches a GpiEndArea. GpiPolygons is not valid inside an area bracket because the figures it defines are areas already.

Although GpiPolygons is limited to straight line figures, it retains a great deal of flexibility. The polygon data structure (POLYGON) contains the number of vertices, and the vertices themselves are in world coordinates. The collection of POLYGON structures accepted in a single GpiPolygons is not limited to one type of polygon.

The drawing of the first polygon begins at the current position. For all subsequent polygons, all vertices must be explicitly defined. If the individual polygons are not completely defined, they are closed with a straight line drawn from the last defined vertex to the first.

After the application has defined the polygons, they may be transformed and manipulated just as other primitives are.

Polygon Boundaries

Applications have two options when specifying the polygon boundary:

- POLYGON_BOUNDARY (the default)
- POLYGON_NOBOUNDARY.

The POLYGON_BOUNDARY value tells the PM programming interface to draw all outlines of the polygon primitives using a line that conforms to the current LINEBUNDLE attributes. If the line attributes have not been changed, the default outline color is black on most display devices and printers, and the default line style is solid. As the attributes cannot be changed within the context of GpiPolygons, all polygons are drawn with the same line. The interior of the polygons are filled with the pattern that conforms to the current AREABUNDLE attributes.

To prevent PM from outlining an area, an application can use the POLYGON_NOBOUNDARY flag when calling GpiPolygons. Only the interior fill pattern is visible. This value is used most often when there are many overlapping polygons, and the interior lines are not desired.

You might think of this option in terms similar to the outline and fill options discussed with boxes and full arcs. The POLYGON_BOUNDARY value corresponds to the DRO_OUTLINEFILL value, and POLYGON_NOBOUNDARY, to the DRO_FILL. There is no distinct boundary value that corresponds to DRO_OUTLINE, the simple outline.

To simulate this effect, OR the POLYGON_NOBOUNDARY value with the appropriate polygon construction value, described below, when calling GpiPolygons.

Polygon Construction

An application specifies the area construction when it calls GpiBeginArea. There are two options:

- POLYGON_ALTERNATE (the default)
- POLYGON_WINDING.

As with area construction, in alternate mode:

- A point is included in the filled polygon if you have to cross an odd number of lines in the set of polygons when drawing a line from that point to infinity.

- A point is not included in the filled polygon if you have to cross an even number of lines in the set of polygons when drawing a line from that point to infinity.

Also as with area construction, in winding mode, the direction in which the boundary lines of the polygons are drawn determines whether a given point is included in the filled polygon. Since the individual polygons drawn with GpiPolygons are generated by independent structures, the direction in which a polygon is drawn depends only on the vertices of that polygon.

To determine if a given point is included in the filled polygon, count the number of lines to be crossed to move from that point to infinity. For each boundary line drawn in one direction add one to the tally. For each line drawn in the opposite direction, subtract one from the tally. A point is within the polygon if the result is nonzero.

Polygon Overlap

An application specifies which pels are drawn when it calls GpiPolygons. There are two options:

- POLYGON_INCL (the default)
- POLYGON_EXCL.

When the overlap value is POLYGON_INCL, the bottom right is included in the polygon. The value POLYGON_EXCL, the exclusive value, indicates that the bottom right is excluded from the polygon. This value acts in conjunction with the mix attribute in determining the appearance of a polygon, which is especially important for a group of overlapping or adjacent polygons.

For example, GpiPolygons specifies a number of polygons. Two of the polygons share the vertex pair (6,7) and (9,7). When the polygons are drawn, if the overlap value was POLYGON_INCL, there are two distinct lines to be drawn from (6,7) to (9,7). A mix attribute other than FM_OVERPAINT could cause undesirable results as the line and drawing-surface colors mix.

As with the polygon boundary and construction values, the overlap value can be ORed when calling GpiPolygons to create a specific effect.

Using Area and Polygon Primitives

You can use area functions to:

- Draw one or more closed figures
- Create a custom fill pattern from a bit map
- Create a custom fill pattern from a font character

Drawing a Single, Closed Figure

The following figure shows an example of how to use area functions to draw a single closed figure that is filled with a vertical pattern using the alternate mode. The closed figure in this example is a 5-pointed star.

```
#define INCL_GPI
#include <os2.h>
void fncAREA02(void){
   POINTL aptl[5];              /* Structure for current position */
   HPS hps;

    /* Initialize the array of points for the 5-pointed star.       */
    aptl[0].x = 400; aptl[0].y = 195;
    aptl[1].x = 40;  aptl[1].y = 320;
    aptl[2].x = 260; aptl[2].y = 10;
    aptl[3].x = 260; aptl[3].y = 390;
    aptl[4].x = 37;  aptl[4].y = 82;

    GpiSetPattern(hps, PATSYM_VERT);/* Set pattern outside bracket */

    /* Draw the star. */
    GpiBeginArea(hps, BA_ALTERNATE);
    GpiMove(hps, &aptl[4]);         /* First and last point of star */
    GpiPolyLine(hps, 5L, aptl);
    GpiEndArea(hps);
} /* fncAREA02 */
```

Drawing Multiple, Intersecting, Closed Figures

The following figure is an example of how to use area functions to draw two intersecting boxes, filled, using the winding mode.

```
#define INCL_GPI
#include <os2.h>
void fncAREA03(void){
    POINTL ptl;            /* Structure for current position */
    HPS    hps;

    GpiBeginArea(hps, BA_WINDING);
    ptl.x = 100;
    ptl.y = 50;
    GpiMove(hps, &ptl);
    ptl.x = 300;
    ptl.y = 250;
    GpiBox(hps, DRO_OUTLINE, &ptl, 0, 0);
    ptl.x = 180;
```

```
    ptl.y = 120;
    GpiMove(hps, &ptl);
    ptl.x = 380;
    ptl.y = 320;
    GpiBox(hps, DRO_OUTLINE, &ptl, 0, 0);
    GpiEndArea(hps);
} /* fncAREA03 */
```

Creating a Custom Fill Pattern from a Bit Map

The following figure is an example of how to create a custom fill pattern by using a hard-coded bit map. In this example, the bit map creates a pattern of arrows.

```
#define INCL_DOS
#define INCL_GPI
#define INCL_WIN
#include <os2.h>
   LONG lcidCustom;                 /* Bit map tag                          */
   HPS  hps;

   VOID CreatePattern(VOID);

   VOID MyFunction(VOID){
       CreatePattern();
       GpiSetPatternSet(hps, lcidCustom);
        .
        .
        .
   } /* func */

   VOID CreatePattern(VOID){
        HBITMAP hbm;               /* Bit map handle                       */
        BITMAPINFOHEADER2 bmp2;    /* Structure for bit map information    */
        PBITMAPINFO2 pbmi2;        /* Pointer to structure for bit map data*/
        PRGB2 prgb2;               /* Structure for color data             */
        ULONG cbBitmapInfo, cColors;

        BYTE abPattern[] = { 0xFF, 0xFF, 0xE7, 0xFF,
                             0xE7, 0xFF, 0xC3, 0xFF,
                             0xC3, 0xFF, 0x81, 0xFF,
                             0x81, 0xFF, 0xE7, 0xFF,
                             0xE7, 0xFF, 0xE7, 0xFF,
                             0xE7, 0xFF, 0xE7, 0xFF,
                             0xE7, 0xFF, 0xE7, 0xFF,
                             0xE7, 0xFF, 0xFF, 0xFF
   };


        lcidCustom = 1;            /* Bit map tag                          */

        bmp2.cbFix = (ULONG) sizeof(BITMAPINFOHEADER2);
        bmp2.cx = 8;               /* Bit map is 8 pels wide               */
        bmp2.cy = 8;               /* Bit map is 8 pels high               */
        bmp2.cPlanes = 1;          /* One bit plane                        */
        bmp2.cBitCount = 1;        /* One bit per pel                      */

        /* Use default values for the remainder of the structure.          */

        bmp2.ulCompression = 0;
        bmp2.cbImage = 0;
        bmp2.cxResolution = 0;
        bmp2.cyResolution = 0;
        bmp2.cclrUsed = 0;
        bmp2.cclrImportant = 0;
        bmp2.usUnits = 0;
        bmp2.usReserved = 0;
        bmp2.usRecording = 0;
        bmp2.usRendering = 0;
```

```
        bmp2.cSize1 = 0;
        bmp2.cSize2 = 0;
        bmp2.ulColorEncoding = 0;
        bmp2.ulIdentifier = 0;

        cColors = 1 << (bmp2.cBitCount * bmp2.cPlanes);

        cbBitmapInfo = sizeof(BITMAPINFO2) + (sizeof(RGB2) * cColors);

        DosAllocMem((PVOID)&pbmi2, cbBitmapInfo,
                    PAG_COMMIT | PAG_READ | PAG_WRITE);

        pbmi2->cbFix = bmp2.cbFix;
        pbmi2->cx = bmp2.cx;
        pbmi2->cy = bmp2.cy;
        pbmi2->cPlanes = bmp2.cPlanes;
        pbmi2->cBitCount = bmp2.cBitCount;

        /* Use default values for the remainder of the structure.            */

        pbmi2->ulCompression = 0;
        pbmi2->cbImage = 0;
        pbmi2->cxResolution = 0;
        pbmi2->cyResolution = 0;
        pbmi2->cclrUsed = 0;
        pbmi2->cclrImportant = 0;
        pbmi2->usUnits = 0;
        pbmi2->usReserved = 0;
        pbmi2->usRecording = 0;
        pbmi2->usRendering = 0;
        pbmi2->cSize1 = 0;
        pbmi2->cSize2 = 0;
        pbmi2->ulColorEncoding = 0;
        pbmi2->ulIdentifier = 0;

        prgb2 = (PRGB2) (pbmi2 + 1);  /* Set address to follow bmp2          */

        /* Set bit map colors to black and white.                            */
        prgb2[0].bBlue = 0;                             /* Color[0] = black */
        prgb2[0].bGreen = 0;                            /* Color[0] = black */
        prgb2[0].bRed = 0;                              /* Color[0] = black */
        prgb2[0].fcOptions = 0;
        prgb2[1].bBlue = 255;                           /* Color[1] = white */
        prgb2[1].bGreen = 255;                          /* Color[1] = white */
        prgb2[1].bRed = 255;                            /* Color[1] = white */
        prgb2[1].fcOptions = 0;

        /* Create a bit map and retrieve its handle.                         */
        hbm = GpiCreateBitmap(hps,
            &bmp2,
            CBM_INIT,
            (PBYTE) abPattern,                          /* Array of bits    */
            pbmi2);

        /* Tag the bit map just created with a custom identifier (lcid).     */
        GpiSetBitmapId(hps, hbm, lcidCustom);
    } /* CreatePattern */
```

Creating a Custom Fill Pattern from a Bit Map

Creating a Custom Fill Pattern from a Font Character

The following figure is an example of how to create a custom fill pattern by using a character from a font. The fill pattern can be only an 8-by-8 bit map; as a result, not all of the character is used.

```
#define INCL_GPI
```

```
#define INCL_WIN
#include <os2.h>
   HPS hps;                     /* Presentation-space handle */
   LONG lcidCustom;             /* Font identifier           */
   FONTMETRICS afm[80];
   FATTRS fat;

   VOID LoadFont(VOID);

   void fncAREA05(void){

        LoadFont();
GpiSetPatternSet(hps, lcidCustom);
GpiSetPattern(hps, 'o');         /* Use lowercase 'o' as fill */
    .
    .
    .
   } /* fncAREA05 */

   VOID LoadFont(VOID){
      LONG cFonts = 0;
      LONG cPublicFonts, i;

      lcidCustom = 1;

/* Determine the number of loaded public fonts.              */
cPublicFonts = GpiQueryFonts(hps, QF_PUBLIC, NULL, (PLONG) &cFonts,
    (LONG) (sizeof(FONTMETRICS)), NULL);

/* Load the metrics for all public fonts into afm.           */
GpiQueryFonts(hps, QF_PUBLIC, NULL, (PLONG) &cPublicFonts,
    (LONG) (sizeof(FONTMETRICS)), afm);

/* Get the first image font with a point size larger than 8.  */
for (i = 0; ((afm[i].fsDefn & FM_DEFN_OUTLINE) ||
     afm[i].lEmHeight <= 8) && i < cPublicFonts; i++);

/* Load the FATTRS structure with the required metrics.      */
fat.usRecordLength = sizeof(fat);
fat.fsSelection = 0;
fat.lMatch = afm[i].lMatch;
StringCopy(fat.szFacename, afm[i].szFacename);
fat.idRegistry = 0;
fat.usCodePage = 0;
fat.lMaxBaselineExt = 0;
fat.lAveCharWidth = 0;
fat.fsType = 0;
fat.fsFontUse = 0;

/* Select this font and assign it a custom lcid.             */
GpiCreateLogFont(hps, NULL, lcidCustom, &fat);
GpiSetCharSet(hps, lcidCustom);
} /* LoadFont */
```

Creating a Custom Fill Pattern from a Font Character

Bit Maps

Raster output devices, such as display screens, are made up of a number of picture elements, called pixels or pels. By setting the color of the pels, you can create an image on the screen. The screen image can be represented internally by a graphics object called a *bit map* . The bit map contains a number of bits that describe the appearance of each pel on the screen.

This chapter describes bit maps, their creation, uses, and functions. The following topics are related to information in this chapter:

- Presentation spaces and device contexts
- Coordinate spaces
- Color and mix modes
- Area primitives
- Paths

About Bit Maps

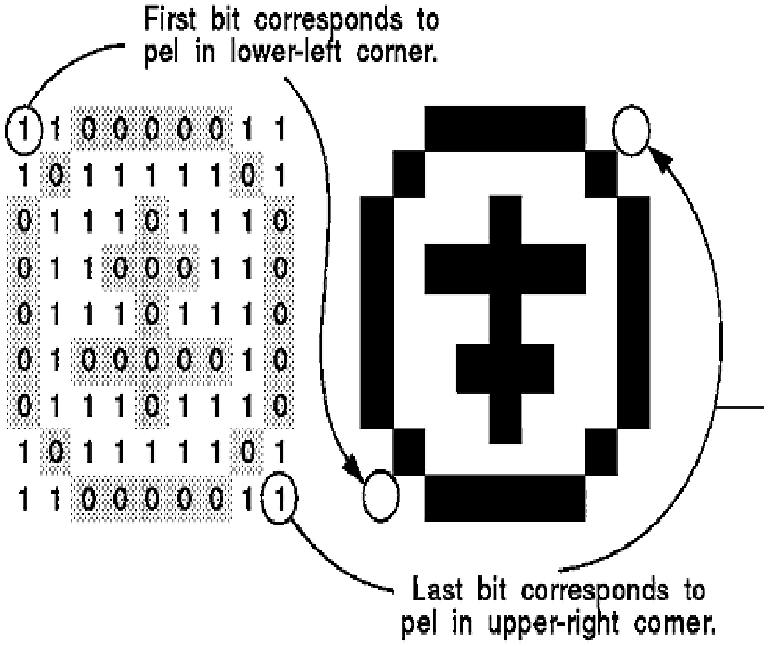
Applications can use bit maps to:

- Store and display scanned images, icons, and symbols
- Create fill patterns for area primitives and paths

An application can display a bit-map image on any *raster* output device. A raster is a rectangular matrix of pels on a video display or dot matrix printer. A raster output device displays an image by setting pels in its matrix to colors specified in a corresponding bit map. An image created in this way is called a "bit-map image". Bit maps cannot be sent to vector output devices such as plotters.

A bit map is drawn to an output device row by row. Each horizontal line of pels is known as a *scan line* .

There is a 1-to-1 correspondence between the number of rows of pels in a bit-map image and the rows of bits in a bit map. The first pel in a bit-map image is in the lower-left corner, and the last pel is in the upper-right corner. The pels are in left-to-right order inside each row of the image. The following figure shows this relationship between bit map and image.



Bits and Pels in a Bit-Map Image

When an application creates a bit map by calling GpiCreateBitmap, it specifies the bit-map width and height in terms of pels in the bit-map image. The width is the number of pels within a row; the height is the number of rows. The application must store these dimensions in the BITMAPINFO2 and BITMAPINFOHEADER and pass their addresses to GpiCreateBitmap.

System Implementation

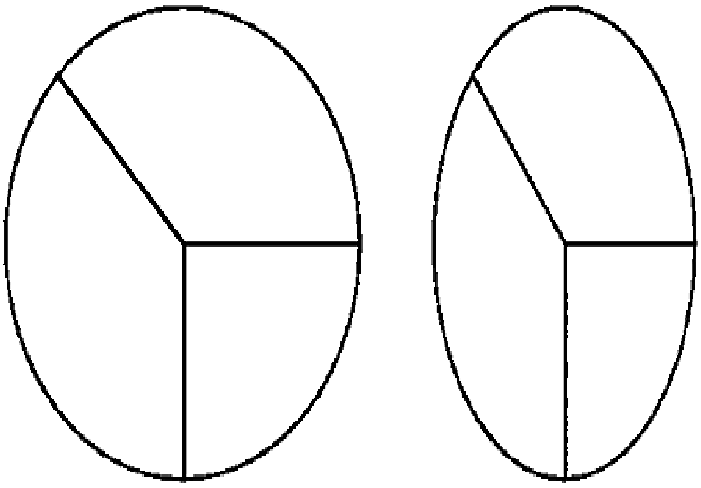
Bit maps are most useful when rapid and frequent movement is required, such as with icons and pointers. They are especially useful for restoring the contents of a window-for example, when an overlying window is removed. If you save the contents of a window in a bit map, you can restore the window contents simply by redisplaying the bit map when the window needs to be redrawn.

Using bit maps is also an effective method of erasing some of the screen contents. For example, you can save the image of the screen in a

bit map at any time while drawing on the screen. If you continue drawing after saving the screen image, you can "erase" any drawing done since you saved the screen image by redisplaying the bit map.

Bit maps are not, however, the recommended way to store graphics that are going to be changed. Most changes to the bit-map contents mean that you have to re-create the bit map.

Bit-map images are device-dependent. Their appearance is affected by the shape of the device's pels and the device's color capabilities. For example, if the pels on one display measure 0.05 mm by 0.1 mm, but 0.1 mm by 0.3 mm on a second display, a circular bit-map pie chart drawn on the first display appears elliptical on the second. The following figure shows how a bit map appears on two types of displays.



Bit Map Shown on Two Types of Displays

Bit maps, particularly color bit maps, can also occupy large amounts of memory. The actual amount of memory occupied by a bit map is determined by both the size of the bit map and the number of bits used to describe each pel.

Bit Map Functions

The OS/2 operating system provides a set of functions that allow you to:

- Create bit maps
- Create and load custom bit maps
- Store color information on a bit map
- Draw bit maps
- Transfer bit maps
- Change the size of a bit map
- Specify the mix values for a bit map
- Convert between monochrome and color data
- Manipulate single pels
- Copy images from a display into a bit map
- Save a bit map
- Delete a bit map
- Make a bit map available to other processes

Creating a Bit Map

A bit map can be created by an application or by using the PM Icon Editor.

By an Application

To enable an application to create a bit map:

1. Create a *memory device context* .

 A memory device context enables an application to treat a bit map in memory as though it were a device. For example, an application can copy color information from another bit map, or copy pels on the display, into a bit map associated with a memory device context.

 To create the memory device context, call DevOpenDC with:

 - A device type of OD_MEMORY (second argument)
 - A handle to a compatible device context (such as the device context of a device with which the bit map is to be compatible).

 Note: The device device-context handle ideally should be the handle of the actual device to which you will be directing the bit map. Otherwise, it will be necessary to change ownership to the appropriate device driver before the BitBlt operation (which copies the bit map from one presentation space to one associated with a screen device context). As a consequence, the image may appear distorted.

 If you omit the handle of the compatible device context by specifying NULL, screen compatibility is assumed.

2. Create a graphics presentation space and associate it with the memory device context.

 The operating system requires this association before the application can perform many of the bit map operations. The handle of this graphics presentation space is required as input to subsequent bit-map-creation and manipulation functions.

3. Create the bit map.

 When an application creates a bit map, the handle of the presentation space that you have associated with the memory device context causes the bit map to be created in a format that is compatible with the memory device context.

 The application also passes two structures: the *bit-map information header* and the *bit-map information table* . These structures contain a great deal of information about the bit map.

 To create the bit map, call GpiCreateBitmap with:

 - The handle of the presentation space (first argument)
 - The bit-map information header, BITMAPINFOHEADER (second argument). (last argument).

 The bit-map information table contains similar information, with the addition of the RGB2 array structure.

 GpiCreateBitmap returns a handle to the bit map, which is used to identify the bit map.

 To determine which bit-map formats are supported on a particular device, call GpiQueryDeviceBitmapFormats. This returns every format supported on a named device. The data is returned as an array of bit-map plane and bit-count pairs. The first pair of values in the array is the one most suitable for the device.

 You can think of the bit map at this stage as a rectangular area of memory containing random data. You can initialize the bit map at this stage by providing GpiCreateBitmap with the address in application storage of some initialization data and by setting the CBM_INIT option. This is a useful function if, for example, your application always starts by displaying the same image.

4. Select the bit map.

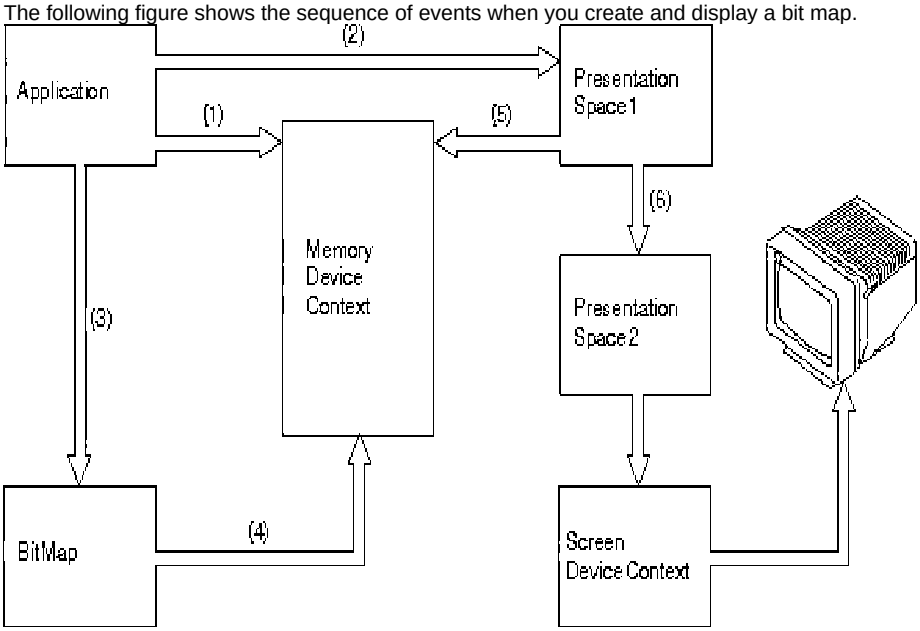
 Before selecting the bit map, you can disassociate the presentation space from the original memory device context and associate it with a different memory device context. However, the bit-map format must be convertible to a format that is supported by the new device. If you have selected one of the four standard bit-map formats, this compatibility is guaranteed and the conversion is automatic.

 Note: When a presentation space is associated with a memory device context, a bit map must be selected into the device context before you can draw in the presentation space.

To select the bit map, call GpiSetBitmap with:

- The presentation-space handle (first parameter)
- The bit-map handle (second parameter).

The following figure shows the sequence of events when you create and display a bit map.



Creating and Displaying Bit Maps

The application:

1. Calls DevOpenDC to create the memory device context.
2. Creates a graphics presentation space. This is associated with the memory device context.
3. Calls GpiCreateBitmap to define a bit map.
4. Calls GpiSetBitmap to designate the bit map as the one currently selected in the memory device context.
5. Calls drawing instructions to the presentation space to draw to the bit map.

 Note: If the bit map is initialized when it is created, this step does not normally exist. Alternatively, this step can be a GpiSetBitmapBits call.
6. Calls GpiBitBlt to copy the bit map from presentation space 1 to presentation space 2 (associated with a screen device context). The bit map is transferred directly to the screen.

Using the Icon Editor

Using the Icon Editor, you can create monochrome or color bit maps that have a *static appearance* . This means that the bit maps can be created in advance and then used without change while the application is running.

When you use the Icon Editor to create a bit map, the bit map is saved in a resource file that can be loaded whenever it is needed. To load a bit-map file, call GpiLoadBitmap, with the identifier of the resource file that contains the bit map, as the second parameter. If you allow this value to default, the application's .EXE file is assumed to contain the bit map.

GpiLoadBitmap lets you specify the x- and y-dimensions (in pels) of the bit map. The loaded bit map is stretched or compressed accordingly. If you supply a 0 value for one of these dimensions, the bit map is sized in the other dimension only, which is likely to cause distortion of the

image. If the bit map is to be produced in its original size, specify 0 for both its width and its height.

Output from the call to this function is the bit-map handle. To display the loaded bit map on the screen, follow the sequence of steps described in the preceding section, omitting steps 3 (defining the bit map) and 5 (issuing drawing instructions to the presentation space).

A bit map created by the Icon Editor is saved in a device-independent format. This format generates an array of bit maps with formats (bits per pel) matching each of the supported display devices.

Creating and Loading a Custom Bit Map

An application can create a custom bit map by setting the bits in an array and passing the array to GpiCreateBitmap or by running the Icon Editor and loading the bit map with GpiLoadBitmap.

To create a custom bit map with an array, an application:

1. Defines an array of bytes that will set pels in an image to the appropriate colors. This array of bytes typically represents the output of a scanned image.
2. Sets the fields in the BITMAPINFOHEADER to their appropriate values.
3. Sets the fields in the BITMAPINFO2 structure to their appropriate values.
4. Calls GpiCreateBitmap, passes it the addresses of the structures and the array of bytes that the application has already defined, and sets the flOptions parameter to CBM_INIT.

If the application is to use this bit map as a fill pattern, it assigns the bit map a local identifier by calling GpiSetBitmapId.

To load a custom bit map that was created with the Icon Editor:

1. Copy the bit map file to the directory in which you compile your application.
2. Create a BITMAP entry in your application's resource file, assigning a unique integer identifier to the bit map.
3. Call GpiLoadBitmap, passing it the identifier that you assigned to the bit map in the resource file.

An application can use GpiLoadBitmap to load any bit map from a file that conforms to any of the standard OS/2 bit-map formats or to a device-specific format supported by the device concerned. This means that an application can load a bit map created by another application, if that application created the correct bit-map header and stored the bit-map bits correctly.

Storing Color Information in a Bit Map

Graphics systems use one of two formats for storing color information in bit maps. The first format uses a single plane and a multiple bit count. The second format uses multiple color planes.

Color Planes

Bit maps are arranged in one or more *color planes*. A color plane is an array of bit-map bits that contains color information.

The bit maps in each of the previous illustrations use the single *bit-map plane* format, which is the standard format for bit maps in OS/2 applications. In this format, a specified number of adjacent bit-map bits contains indexes to either a special color table of RGB values or actual RGB2 structures. Whether the application maps bits into an RGB or RGB2 structure depends on the bit-map format. All of the color information resides in a single plane.

The second color format uses more than one color plane. A common multiplane bit-map format is the three-plane format, in which one plane corresponds to the red pels, another to the green pels, and a third to the blue pels. Multiplane bit-map formats are rare in PM applications.

Most bit maps are stored externally in a single-plane format, although the device driver (such as VGA) may internally convert them to the multiplane format.

The single-plane format can be converted internally to any multiplane format used by a device. You also can use a nonstandard number of bits to describe each pel, if supported by your output device. If you write your own presentation driver, it must be able to convert the standard bit-map formats to its own internal format.

An application can determine which color-plane format a device supports by calling GpiQueryDeviceBitmapFormats.

Standard Bit-Map Formats

On a monochrome device, you need only one bit to describe a single pel, and that bit is switched on or off. Color devices require more bits. For example, an eight-color picture requires three bits to describe a single pel, because each component of the RGB mix (red, green, blue) that gives a pel its color must be described.

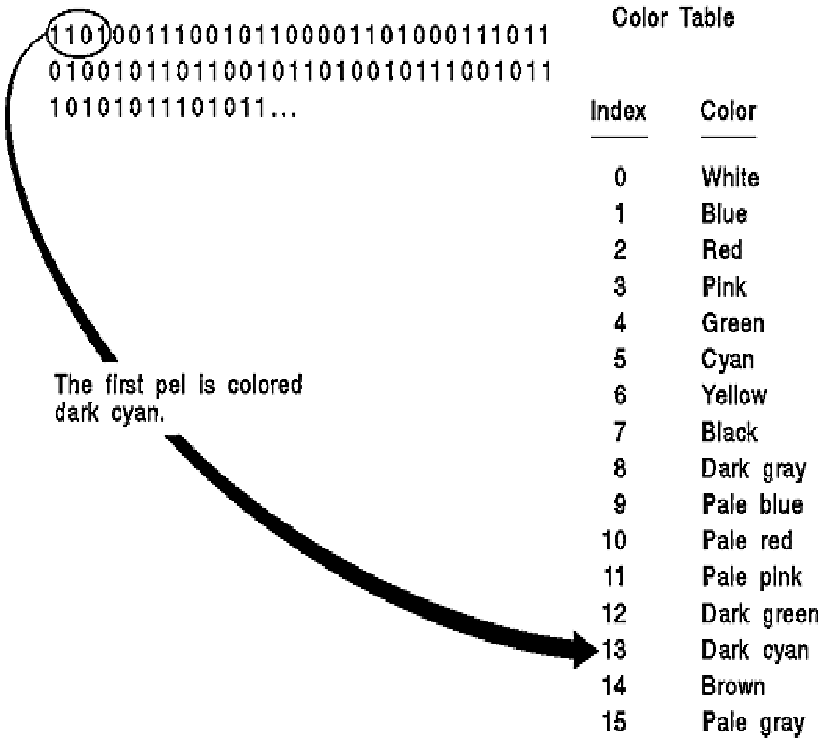
A bit count is a value that specifies how many adjacent bit-map bits correspond to each pel in a bit-map image. There are four standard bit-map formats, each with a different bit count. The formats are shown in the following table.

Format	Bits per pel	Size of 640 x 480 image in bytes (uncompressed)
Monochrome	1	38 400
16 color	4	153 600
256 color	8	307 200
16.7 million color	24	921 600

Note: The bits are stored consecutively in the bit-map plane. If you have four bits for each pel, the four bits for pel 1 are followed by the four bits for pel 2, and so on. The bits that describe pel 1 are stored beginning in the most-significant bits of the first byte. The data for each scan line is packed together, and the bottom scan line appears first in memory with the leftmost pel first. Each scan line, however, is padded at the end so that each line begins on a ULONG (32-bit) boundary.

If the device supports a bit count of one bit per pel, the color table contains two entries. A device that supports a bit count of *n* bits per pel, has a corresponding color table with 2n entries. However, a bit count of 24 bits per pel indicates that there is no color table, because each pel is a direct RGB value.

The following figure shows a bit map using a bit count of four bits per pel and an associated color table:



A Bit Map and Its Associated Color Table

If a device uses a bit count of one, four or eight bits per pel, the bit-map bits contain index values for a bit-map color table. If the device supports a bit count of 24 bits per pel, the bit-map bits contain the bRed, bGreen, and bBlue fields of RGB2 structures. No color table is associated with a bit map on a device that supports a format of 24 bits per pel-such a device can support over 16 million colors. Instead of using a color table, the BITMAPINFO2 structure consists of only the header, and the red, green, and blue color values are provided directly by the bit-map data.

An application can determine the bit-count format that a device supports by calling GpiQueryDeviceBitmapFormats.

Drawing Bit Maps

An application can draw bit-map images on a raster printer or display screen, or into metafiles associated with a raster device. Any GPI drawing requests (including those that produce graphics text), issued to a presentation space associated with a memory DC containing a selected bit map, cause the bit map to receive raster images of your drawings. The following table describes the bit map drawing functions:

Function	Input	Output
WinDrawBitmap	The handle of a bit map.	A bit-map image on a raster display.
GpiImage	A buffer containing bit map image data.	A special monochrome bit-map image on a raster display or printer.
GpiDrawBits	A buffer containing bit map image data.	A bit-map image on a raster display or printer.

GpiBitBlt	The handle of a presentation space containing a bit map.	A bit-map image on a raster display or printer, or a bit-map image into a metafile (albeit in an escape order).
GpiWCBitBlt	The handle of a bit map.	A bit-map image on a raster display or printer, or a bit-map image into a metafile.

WinDrawBitmap

WinDrawBitmap draws a bit-map image by copying it into a window linked to a target presentation space. A call to this function is valid only in draw mode (DM_DRAW), and only for a screen device context. This function does not require an application to select a bit map into a presentation space before the application draws the corresponding image. An application can use WinDrawBitmap to scale bit maps by specifying DBM_STRETCH as the last argument, and the address of a RECTL structure as the fourth argument. The coordinates in this structure are always device coordinates.

WinDrawBitmap draws both color and monochrome bit maps. Color bit maps require no color conversion. Monochrome bit maps can be drawn in any two colors which can be explicitly specified as parameters to the call or taken from the image bundle. These parameters will be color table indexes or RGB values, depending on the color table mode of the target presentation space. The current image bundle mix modes are used and, for certain mix values, will affect color.

You can call WinDrawBitmap in retain mode, but the bit-map image will only be drawn and not recorded in the segments.

Note: An application can determine the current colors and their corresponding mix modes by calling GpiQueryAttrs. The application can set them by calling GpiSetAttrs.

An *inverted* bit map is a bit map in which the colors have been inverted; white becomes black and black becomes white. An application can draw inverted bit maps by calling WinDrawBitmap and passing it DBM_INVERT as the last argument. An application can draw halftone bit maps by calling WinDrawBitmap and passing it DBM_HALFTONE as the last argument. Before drawing the bit map, clear the presentation space to CLR_BACKGROUND using GpiErase.

GpiImage

GpiImage draws a nonstandard, monochrome (two-color) bit map called an *image primitive* . The bit-map bits in an image are stored in the opposite order from the bits in a standard bit map-the first bit in the bit map corresponds to the pel in the upper-left corner of the bit-map image, and the last bit in the bit map corresponds to the pel in the lower-right corner of the bit-map image. The following figure shows the correspondence between the bits in an image primitive and the pels in the drawing produced by GpiImage.

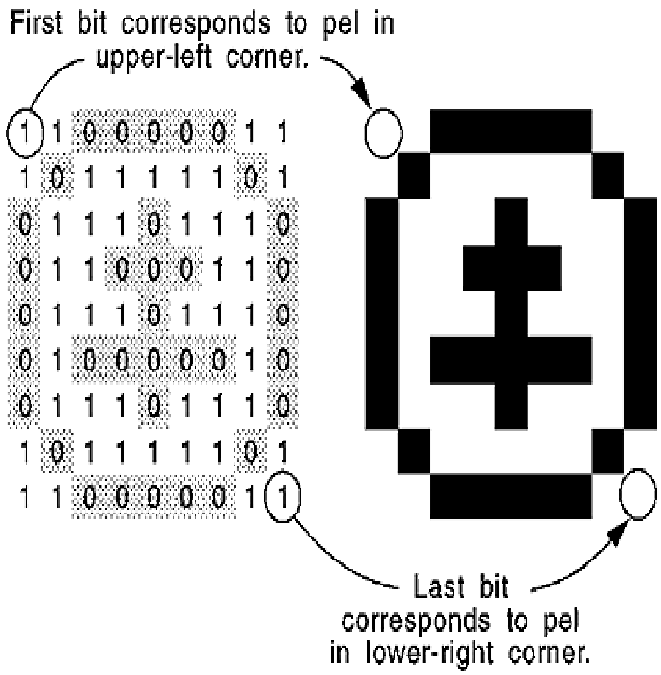


Image-Primitive Bits and Pels

A call to GpiImage also is valid only in draw mode (DM_DRAW), but can provide output to a screen or printer. An application cannot scale bit-map images by using GpiImage.

Transferring Bit Maps

The three remaining bit-map drawing functions operate in very similar ways. GpiDrawBits copies a bit-map image from application memory to a device or a device context. GpiBitBlt directs bit maps to devices other than the screen. GpiWCBitBlt enables you to retain the bit-map data in the segment store of the target presentation space.

The similarities are discussed first. Their differences are discussed in the sections that follow.

An application should use GpiDrawBits, GpiBitBlt, or GpiWCBitBlt to draw bit maps that use a color table or RGB2 structures color formats of 1, 4, 8, or 24 bits per pel.

An application can draw inverted bit maps for any of these three functions by calling the function and passing ROP_NOTSRCCOPY as the raster operation.

GpiDrawBits

GpiDrawBits copies bit map image data from storage into a bit map that has been selected into a device context associated with a presentation space. It can also copy bit-map image data to a device. An application can use this function to draw a bit map without first selecting the bit map into a presentation space.

This function is valid in all draw modes. Set the draw mode to DM_RETAIN or DM_DRAWANDRETAIN to create retained segments; otherwise, the default mode, DM_DRAW, is selected.

GpiBitBlt

GpiBitBlt requires an application to use device coordinates for the dimensions of the source and target rectangles. GpiBitBlt allows you to direct bit maps to devices other than the screen. It is independent of the drawing mode, but it operates as if in draw mode.

At its simplest, GpiBitBlt copies a whole bit map, unaltered, from a device or bit map source to a device or bit map target. At its most powerful, GpiBitBlt can take a part of or a whole bit map, and alter both its size and appearance in the process of copying it to another device context. A bit map can be copied:

- From one memory device context to another memory device context
- From a memory device context to the device context of an output device (screen window or printer) that supports raster operations
- From the device context of an output device to a memory device context
- From the device context of an output device to another device context of the same output device. In both cases, the screen device can be a PM window.

A memory device context (whether it is the source or the target of the GpiBitBlt operation) *must* have a bit map selected when GpiBitBlt is called.

GpiBitBlt has a number of input parameters, two of which are the handles of two presentation spaces: the source presentation space and the target presentation space. Both of these presentation spaces must be associated with an appropriate device context. Unless the associated device is a banded printer, a single presentation space can be both source and target. This allows you to copy a bit map within a single PM window, or to update the bit map. The source and target rectangles are specified in device coordinates in GpiBitBlt. GpiBitBlt, consequently, is very device-dependent, and should be avoided when creating data for interchange.

GpiWCBitBlt

GpiWCBitBlt enables you to retain the bit-map data in the segment store of the target presentation space. It is a valid in all three drawing modes:

- DM_DRAW-display, printer, or into metafile
- DM_RETAIN-into metafile or segment
- DM_DRAWANDRETAIN-display, printer, then into associated segment or metafile.

Most of the bit-map drawing operations occur in an application's device space. GpiWCBitBlt, however, lets an application draw in its world space, but requires that you use device coordinates for the source rectangle, and world coordinates for the target rectangle.

An application can use GpiWCBitBlt to draw a bit map with consistent dimensions on devices with different *aspect ratios* . The aspect ratio is the ratio of a pel's width to its height.

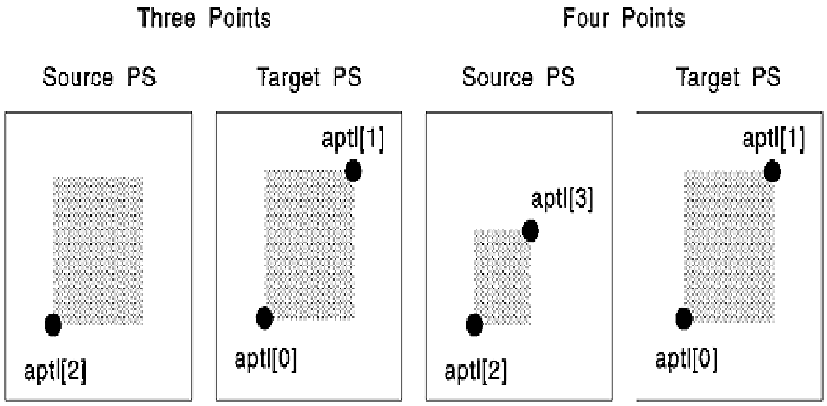
When creating data for interchange, use GpiWCBitBlt. GpiWCBitBlt provides the same function as GpiBitBlt, with the following exceptions:

- The target rectangle is specified in world coordinates, and all four coordinates (the two source-rectangle coordinates and the two target-rectangle coordinates) must be specified.
- The source handle must be a bit-map handle. It must not be the handle of a source presentation space. The bit map identified by the source handle must not be selected into a memory device context when you call GpiWCBitBlt.
- GpiWCBitBlt conforms to the current drawing mode in the target presentation space. If the drawing mode is retain or draw-and-retain, the bit map is retained in segment store.

Changing the Size of the Bit Map

You can specify the sizes of the rectangular bit blocks in both the source and the target presentation spaces. To do this, provide an array of up to four device-coordinate positions as input to the call. The first two positions define the lower-left and upper-right corners of the target rectangle; the second two define the same two corners of the source rectangle.

If you want the two rectangles to be of equal size, do not specify the device coordinates of the upper-right corner of the source rectangle. The correct amount of data is automatically transferred to fill the target rectangle. The following figure shows the points count for bit-block transfers.



Points Count for Bit-Block Transfers

Equal-size rectangles can be *built* much faster than rectangles that need stretching or compressing. Compression options (flOptions) are ignored if the rectangles are to be of equal size.

If the rectangles are not to be of equal size, you must specify all four coordinate points. The bit-map data is stretched, if the target rectangle is larger than the source rectangle, or compressed, if the target rectangle is smaller, to fit the target rectangle. The bit map is stretched by duplicating rows and columns of data, an action that might cause distortion of the image. If the data is to be compressed, you can specify one of three compression options as shown in the following table.

Bit-map Data Compression Rules

Option	Compression Rules
BBO_OR	Compresses the bit map data as necessary, using a logical OR operation on the eliminated rows and columns. This is useful for preserving the foreground when foreground pels are "1" and the background pels are "0".
BBO_AND	Compresses the bit map data as necessary, using a logical AND operation on the eliminated rows and columns. This is useful for preserving the foreground when foreground pels are "0" and the background pels are "1".
BBO_IGNORE	Compresses the bit map data as necessary, but ignores any eliminated rows or columns. This is most useful for color bit maps, where the results of combining pels of different colors are unpredictable.

Specifying Mix Values

When you draw a graphics primitive into a presentation space, it is affected by the current foreground-mix value, and possibly by the current background-mix value, in that presentation space. However, when you copy a bit map into a target presentation space, the current foreground- and background-mix values in the target presentation space are ignored. Instead, you specify a mix-mode value, also known as the *raster-operation* (ROP) value. This value determines:

- How the bit map is affected by any data that might already be in the target presentation space
- The color of each pel in the bit map

Each pel in the bit map of the target presentation space has, potentially, three color settings:

- The setting of that pel in the source rectangle
- The initial setting of that pel in the target rectangle
- The setting of that pel in the current area-fill pattern in the target presentation space

The (boolean) values of each of these settings can be combined using boolean operations to produce the final value of each pel in the target presentation space. For a color target, the target must be regarded as consisting of multiple one-bit per pel planes, with the mixing applied to each plane separately to produce the final color index into the physical color table. Because the final color index is an index into the physical palette, the results of any color mixing are therefore unpredictable. For example, if you ORed together two numeric index values, the color indexed by the result is unlikely to have any direct relation to the colors indexed by the two values you ORed together. It depends on the order of the colors in the physical table.

As input to the functions, you supply an actual mix value. The ROP mix value required to achieve any given result can be determined from the following table. The final value of each bit in every pel depends on the values of the corresponding bits in the pattern (P), source (S), and the original target value (T initial). Each row of the table shows one of the eight possible combinations of these values. For each combination, mark the desired final target value in the last column. The eight bits in this column then show the value of the least significant byte of the ROP value required to achieve this mixing function. For example, if the required mixing function is to copy the source to the target, then the target (final) column will be the same as the S column, and the ROP value will have the binary value 11001100, or the hexadecimal value 00CC.

Possible Settings for the Index Bits

Pattern	Source	Target (initial)	Target (final)
0	0	0	Index bit 0
0	0	1	Index bit 1
0	1	0	Index bit 2
0	1	1	Index bit 3
1	0	0	Index bit 4
1	0	1	Index bit 5
1	1	0	Index bit 6
1	1	1	Index bit 7

If you want the pattern to have no effect on the target rectangle, the four low-order bits of the index must be the same as the four high-order bits.

A monochrome bit map consists only of source settings of 1s and 0s, where a 1 or 0 represents each pel. It also has a two-entry color table with index 0 specifying what color a 0 pel represents and index 1 representing what color a 1 pel represents. In a monochrome bit map, this color table is not used during BitBlt operations (but is used when drawing to the bit map and moving the bit map between devices). The preceding table defines how any combination of pattern, source and target original (1 and 0) values define the target final (1 or 0) value. A color bit map has an index (in the case of four bits, 0-15) for each pel which, serves as index into the bit map color table (16 entries for four bits per pel) to define the color of each pel. Unlike monochrome bit maps, this table is used during BitBlt operations (instead of target image bundle attributes) to provide the color of the target output.

The following table represents a selection of the most commonly used mixes with predefined symbolic names and their effects. (There are many other possible mixes).

Mix Value Names

Mix Name	Effect
ROP_SRCCOPY	Source
ROP_SRCPAINT	Source OR Target
ROP_SRCAND	Source AND Target
ROP_SRCINVERT	Source XOR Target
ROP_SRCERASE	Source AND NOT (Target)
ROP_NOTSRCCOPY	NOT (Source)
ROP_NOTSRCERASE	NOT (Source) AND NOT (Target)
ROP_MERGECOPY	Source AND Pattern
ROP_MERGEPAINT	NOT (Source) OR Target
ROP_PATCOPY	Pattern
ROP_PATPAINT	NOT (Source) OR Pattern OR Target
ROP_PATINVERT	Target XOR Pattern
ROP_DSTINVERT	NOT (Target)
ROP_ZERO	0
ROP_ONE	1
ROP_GRAY	Gray pattern background

ROP_SRCCOPY is the most often used value. It simply copies the bit map to the target presentation space without performing any color mixing. Because most of the remaining mixes involve some degree of color mixing, they result in an image that is quite different from the original.

Converting between Monochrome and Color Data Bit Maps

It is possible to copy a monochrome bit map to a color bit map or device, and to copy a color bit map to a monochrome bit map or device. PM handles these conversions automatically according to the following rules:

- If you are copying a monochrome bit map to a color bit map or to a color or monochrome device surface, a source setting of 1 adopts the current image-foreground color of the target presentation space, and a source setting of 0 adopts the current image-background color of the target presentation space. For example, if the image foreground color is blue and the image background color is yellow in the target presentation space, a monochrome bit map is converted to a blue foreground on a yellow background.
- If you are copying from a monochrome pattern to a color bit map or device, a source setting of 1 adopts the current area-foreground color of the target device and a source setting of 0 adopts the current area-background color of the target device. Note that if image bundle attributes do not exist for the source bit map, as is the case when using GpiWCBitBlt (which uses a bit map handle as the source), then zero source pixels adopt the image background color and nonzero source pixels adopt the image-foreground color of the presentation space.
- If you are copying a color bit map to a monochrome bit map or device, those pels that have the same color as the current image-background color in the source presentation space adopt the image background color of the target presentation space. For example, if the current image-background color in the source presentation space is blue, all blue pels in the bit map take on the color of the current image background in the target presentation space.

All other pels in the color bit map adopt the current image foreground color of the target presentation space.

Manipulating Single Pels

You can manipulate single pels in a bit map by using GpiSetPel and GpiQueryPel. GpiSetPel sets the pel at the specified point (in world coordinates) to the current line color. This function is independent of the current drawing mode, and its effect is immediate. It is, however, a device-dependent function. The CAPS_RASTER_CAPS option of DevQueryCaps indicates if GpiSetPel is supported on the current device.

Copying Images from a Display into a Bit Map

An application can copy an image from a raster display screen to a bit map by calling GpiBitBlt or GpiWCBitBlt. Before copying the bit map, the application must call DevOpenDC to create a memory device context. This device context allows an application to treat a bit map in memory as though it were a device-the application can copy color information from pels on the display to the bit map.

Once an application creates a memory device context, associates it with a presentation space, and selects a bit map into the presentation space, the application can use the presentation-space handle as the first argument to GpiBitBlt or GpiWCBitBlt. If the application will be drawing the image on devices with different aspect ratios, the application should use GpiWCBitBlt to preserve the original dimensions of the bit map.

Saving a Bit Map

You have two ways to save a bit map that has been created by an application:

- Store the bit map in a *metafile* , by calling GpiWCBitBlt.

 Metafiles are covered in detail later.

- Use the IBM OS/2 file-handling functions, in conjunction with GpiQueryBitmapBits and GpiSetBitmapBits.

 An application can save a bit map in a file by calling GpiQueryBitmapBits, DosOpen, DosWrite, and DosClose.

GpiQueryBitmapBits copies bit-map data from a memory device context (that has a standard-format bit map selected into it) to a buffer. The bit map you copy can be newly created or have been loaded from a resource file.

You must supply the address in application storage of a *bit-map information table* . The bit-map information table has the same structure as the bit map information header, except for an additional entry, the colors field. As input to GpiQueryBitmapBits, you supply a bit-count value and a plane value. These must both be standard bit-map format values, so the plane value is always 1, and the bit-count value is 1, 4, 8, or 24. If any conversion of the bit-map data is necessary, it is done automatically. You also must provide the address of the application storage into which the bit-map data is to be loaded. You probably will need to find out how large the bit map is before you can do this. GpiQueryBitmapParameters returns the width, height, plane count, and bit count of a named bit map.

You also must supply the size of the fixed portion of the header (as well as bit count, and so on). Nonstandard formats supported by the device that owns the bit map will also be valid for GpiQueryBitmapBits.

You must ensure that you allocate sufficient storage at the end of the bit map information table for the returned color table which, if the number of planes is 1, will have 2n entries, where n is the number of bits per pel.

On return from GpiQueryBitmapBits, the system supplies the width, height, and bit-map color table in the bit-map information table. You can retrieve a part of a bit map by specifying the number of the scan line from which the transfer is to start, and the number of scan lines you want.

When you have copied a bit map into application storage, you can save the bit map externally, and reload it later, by using OS/2 file-handling functions. After the application creates a file by calling DosOpen, it can call DosWrite to copy the buffer containing the bit-map bits into the file. The application then closes the file by calling DosClose.

If an application needs to use a bit map it has stored on disk, it can copy the file's contents into a buffer by calling DosRead, and then copy information about the image to the bit map by calling GpiSetBitmapBits. The application must select the bit map into a memory device context before it sets the bits. As with GpiQueryBitmapBits, you can transfer a part of the bit map rather than the whole. GpiSetBitmapBits can be used to transfer standard-format bit maps only, and only to a bit map selected into a memory device context. If necessary, bit-map data is automatically converted from one standard format to another. Nonstandard formats supported by the device owning the bit map will also be valid for GpiSetBitmapBits.

If an application creates bit maps another application might use, it should create them by using the standard IBM OS/2 bit-map file format.

Deleting a Bit Map

It is good practice always to delete a bit map from memory when you have finished using it. To delete a bit map, use GpiDeleteBitmap. If you have loaded a bit map from a resource file, GpiDeleteBitmap deletes only its memory version. If the bit map has not been deleted when the owning process ends, it is deleted automatically by the system.

Making Bit Maps Available to Other Processes

When an application creates a bit map or loads one from a resource file, it can make the bit map available to other processes by placing the bit-map handle in the clipboard.

Using Bit Maps

An application can use bit-map functions to:

- Copy an image from a raster device (such as a display screen) to a bit map
- Copy an image from a raster device (such as a display screen) to the same device
- Copy an image from a bit map to a raster device
- Copy an image from a bit map to another bit map
- Copy an image from application memory to a bit map
- Copy an image from application memory to a raster device
- Scale bit-map images
- Create custom fill patterns for area primitives and paths
- Load a bit map created with the Icon Editor
- Draw bit-map images
- Store bit maps in a metafile or application memory

Copying an Image from a Display Screen to a Bit Map

To copy an image from a display screen to a bit map:

1. Associate the memory device context with a presentation space.

2. Create a bit map.
3. Select the bit map into the memory device context by calling GpiSetBitmap.
4. Determine the location (in device coordinates) of the image.
5. Call GpiBitBlt and copy the image to the bit map.

The following figure demonstrates these steps.

```
HDC hdcMem;
PSZ pszData[4] = { "Display", NULL, NULL, NULL };
HPS hpsMem, hps;
HAB hab;
SIZEL sizlPage = {0, 0};
BITMAPINFOHEADER2 bmp;
PBITMAPINFO2 pbmi;
HBITMAP hbm;
SHORT sWidth = 128, sHeight = 128;
POINTL aptl[3];
LONG alData[2];

/*
 * Create the memory device context and presentation space so they
 * are compatible with the screen device context and presentation space.
 */

hdcMem = DevOpenDC(hab, OD_MEMORY, "*", 4,
    (PDEVOPENDATA) pszData, NULLHANDLE);

hpsMem = GpiCreatePS(hab, hdcMem, &sizlPage,
    PU_PELS | GPIA_ASSOC | GPIT_MICRO);

/* Determine the device's plane/bit-count format. */

GpiQueryDeviceBitmapFormats(hpsMem, 2, alData);

/*
 * Load the BITMAPINFOHEADER2 and BITMAPINFO2 structures. The sWidth and
 * sHeight fields specify the width and height of the destination
 * rectangle.
 */

bmp.cbFix = (ULONG) sizeof(BITMAPINFOHEADER2);
bmp.cx = sWidth;
bmp.cy = sHeight;
bmp.cPlanes = alData[0];
bmp.cBitCount = alData[1];
bmp.ulCompression = BCA_UNCOMP;
bmp.cbImage = (((sWidth *
    (1 << bmp.cPlanes) * (1 << bmp.cBitCount)) + 31) / 32) * sHeight;
bmp.cxResolution = 70;
bmp.cyResolution = 70;
bmp.cclrUsed = 2;
bmp.cclrImportant = 0;
bmp.usUnits = BRU_METRIC;
bmp.usReserved = 0;
bmp.usRecording = BRA_BOTTOMUP;
bmp.usRendering = BRH_NOTHALFTONED;
bmp.cSize1 = 0;
bmp.cSize2 = 0;
bmp.ulColorEncoding = BCE_RGB;
bmp.ulIdentifier = 0;


DosAllocMem((PPVOID)&pbmi, sizeof(BITMAPINFO2) +
    (sizeof(RGB2) * (1 << bmp.cPlanes) * (1 << bmp.cBitCount)),
    PAG_COMMIT | PAG_READ | PAG_WRITE);

pbmi->cbFix = bmp.cbFix;
pbmi->cx = bmp.cx;
pbmi->cy = bmp.cy;
pbmi->cPlanes = bmp.cPlanes;
pbmi->cBitCount = bmp.cBitCount;
pbmi->ulCompression = BCA_UNCOMP;
pbmi->cbImage = ((sWidth+31)/32) * sHeight;
```

```
pbmi->cxResolution = 70;
pbmi->cyResolution = 70;
pbmi->cclrUsed = 2;
pbmi->cclrImportant = 0;
pbmi->usUnits = BRU_METRIC;
pbmi->usReserved = 0;
pbmi->usRecording = BRA_BOTTOMUP;
pbmi->usRendering = BRH_NOTHALFTONED;
pbmi->cSize1 = 0;
pbmi->cSize2 = 0;
pbmi->ulColorEncoding = BCE_RGB;
pbmi->ulIdentifier = 0;

/* Create a bit map that is compatible with the display.            */

hbm = GpiCreateBitmap(hpsMem, &bmp, FALSE, NULL, pbmi);

/* Associate the bit map and the memory presentation space.         */

GpiSetBitmap(hpsMem, hbm);

/* Copy the screen to the bit map.                                  */

aptl[0].x = 0;       /* Lower-left corner of destination rectangle  */
aptl[0].y = 0;       /* Lower-left corner of destination rectangle  */
aptl[1].x = sWidth;  /* Upper-right corner of destination rectangle */
aptl[1].y = sHeight; /* Upper-right corner of destination rectangle */
aptl[2].x = 0;       /* Lower-left corner of source rectangle       */
aptl[2].y = 0;       /* Lower-left corner of source rectangle       */

hps = WinGetScreenPS(HWND_DESKTOP);

GpiBitBlt(hpsMem, hps,
    sizeof(aptl) / sizeof(POINTL), /* Number of points in aptl      */
    aptl, ROP_SRCCOPY, BBO_IGNORE);

WinReleasePS(hps);
```

Scaling and Drawing a Bit-Map Image

You can scale a bit map by calling GpiBitBlt or GpiWCBitBlt and altering the dimensions of the target rectangle. The following figure shows how to shrink the screen copied in the first example to half its original size, and then redraw it by calling GpiBitBlt.

```
POINTL aptl[3];
HPS hpsMem, hps;
HWND hwnd;
SHORT sWidth = 128, sHeight = 128;
/* Target-rectangle dimensions (in device coordinates)              */
aptl[0].x = 0;
aptl[0].y = 0;
aptl[1].x = sWidth / 2;
aptl[1].y = sHeight / 2;

/* Source-rectangle dimensions (in device coordinates)              */
aptl[2].x = 0;
aptl[2].y = 0;
aptl[3].x = sWidth;
aptl[3].y = sHeight;

hps = WinGetPS(hwnd);

GpiBitBlt(hps, hpsMem,
    sizeof(aptl) / sizeof(POINTL),      /* Number of points in aptl */
    aptl, ROP_SRCCOPY, BBO_IGNORE);

WinReleasePS(hps);
```

Creating a Custom Fill Pattern

To create a custom fill pattern that the operating system will use to fill area primitives and paths:

1. Set an array of bits for a bit map that measures 8-bits-by-8-bits (remember that the operating system pads the bit-map bits on a ULONG (32-bit) boundary).
2. Create a bit map in a screen presentation space by calling GpiCreateBitmap and passing it the address of the array of bits from Step 1.
3. Assign a local identifier (lcid) to the bit map by calling GpiSetBitmapId.
4. Set the attribute of the pattern set in the AREABUNDLE structure by calling GpiSetPattern.

The following figure shows how to create the pattern.

```
/* Define an array of bytes;  this array creates a grid pattern. */

BYTE abPattern5[] = {
    0xFF, 0xFF, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00,
    0x80, 0x00, 0x00, 0x00 };

    LONG lcidCustom = 1;
HPS hps;
PBITMAPINFO2 pbmi;
BITMAPINFOHEADER2 bmp;
HBITMAP hbm;
PRGB2 prgb2;

/* Create the bit map, passing the address of the array of bytes. */
hbm = GpiCreateBitmap(hps, &bmp, CBM_INIT, (PBYTE) abPattern5, pbmi);

/* Assign a local identifier to the bit map.                      */
GpiSetBitmapId(hps, hbm, lcidCustom);

/* Set the pattern-set attribute in the AREABUNDLE structure.     */
GpiSetPatternSet(hps, lcidCustom);
```

Loading a Bit Map from a File

You can load a bit map from a file if the format of the file corresponds to the standard IBM OS/2 bit-map file format. (Any bit map that you create with the Icon Editor is automatically stored in this format.) To load a bit map:

1. Copy the bit-map file to the directory that contains your application's resource file and source code.
2. Create an entry in your application's resource file, assigning a unique integer identifier to the bit map.

3. Call GpiLoadBitmap in your application's source code, passing it the integer identifier that you assigned to the bit map in your application's resource file.

You can actually include your bit map in the application's .EXE file or in a separate resource file. If the bit map is included in a separate resource file, you must specify both the resource ID and the ID of the bit map within the resource file on GpiLoadBitmap. If bit map is to be included in the application's .EXE file, the resource ID is specified as NULL and only the ID of the bit map is required by GpiLoadBitmap.

Following is an example of code from an application's resource file that assigns the integer value 200 to a bit-map file called CUSTOM.BMP.

```
BITMAP  200 CUSTOM.BMP
```

The following figure is an example of code from the application that shows how to retrieve a bit-map handle by calling GpiLoadBitmap, use the handle to tag the bit map by calling GpiSetBitmapId, and then use the local identifier supplied by GpiSetBitmapId to set the bit map as the current fill pattern, using GpiSetPatternSet.

```
HPS hps;
HBITMAP hbm;
LONG lcidCustom = 1;
POINTL ptl;

hbm = GpiLoadBitmap(hps,  /* Presentation-space handle        */
    NULLHANDLE,           /* Resource in application's module */
    IDB_PATTERN,          /* Bit-map ID                       */
    16,                   /* Bit-map width                    */
    16);                  /* Bit-map height                   */

/* Assign a local identifier to the bit map.                  */
GpiSetBitmapId(hps, hbm, lcidCustom);

/* Set the pattern-set attribute in the AREABUNDLE structure. */
GpiSetPatternSet(hps, lcidCustom);

ptl.x = 100;
ptl.y = 100;
GpiMove(hps, &ptl);
ptl.x = 200;
ptl.y = 200;
GpiBox(hps, DRO_OUTLINEFILL, &ptl, 0, 0);
```

Storing a Bit Map in a Metafile

You can draw bit maps in a metafile or segment by calling GpiWCBitBlt. The operating system converts this function to a drawing order. The target-rectangle dimensions that you pass to GpiWCBitBlt are in world coordinates, not device coordinates. The following figure shows how to draw a bit map in a metafile, and then play the metafile.

```
DEVOPENSTRUC dop;
HDC hdcMeta;
HPS hps, hpsMeta;
SIZEL sizlPage;
HMF hmf;
HBITMAP hbm;
HAB hab;
HWND hwnd;
POINTL aptl[4];

dop.pszLogAddress = NULL;
dop.pszDriverName = "DISPLAY";
dop.pdriv = NULL;
dop.pszDataType = NULL;

hdcMeta = DevOpenDC(hab, OD_METAFILE, "*", 4L,
```

```
              (PDEVOPENDATA) &dop, NULLHANDLE);
hpsMeta = GpiCreatePS(hab, hdcMeta, &sizlPage, PU_PELS | GPIA_ASSOC);

hbm = GpiLoadBitmap(hpsMeta, NULLHANDLE, IDB_PATTERN, 16L, 16L);

aptl[0].x = aptl[0].y = 0;  /* Lower-left corner target rectangle        */
aptl[1].x = 150;            /* X coordinate upper-right target rectangle */
aptl[1].y = 300;            /* Y coordinate upper-right target rectangle */
aptl[2].x = aptl[2].y = 0;  /* Lower-left corner source rectangle        */
aptl[3].x = aptl[3].y = 16; /* Upper-right corner source rectangle       */

GpiWCBitBlt(hpsMeta, hbm, 4L, aptl, ROP_SRCCOPY, BBO_IGNORE);

GpiAssociate(hpsMeta, NULLHANDLE);
hmf = DevCloseDC(hdcMeta);

hps = WinGetPS(hwnd);

GpiPlayMetaFile(hps, hmf, 0L, NULL, NULL, 0L, NULL);

WinReleasePS(hps);
```

You can also store a bit map in a metafile by calling GpiBitBlt. In this case, however, the bit map will be stored inside an escape order.

Creating and Drawing Retained Graphics

There are two types of graphics output in the OS/2 operating system:

- *Retained graphics* , which are stored in segments and can be redrawn or edited when necessary
- *Nonretained graphics* , which are drawn immediately but not stored and therefore cannot be used again without repeating the graphics functions needed to re-create the picture

This chapter describes the advantages of using retained graphics and provides information on creating and drawing retained graphics.

The following topics are related to the information in this chapter:

- Presentation spaces and device contexts
- Coordinate spaces and transformations
- Editing retained graphics and graphics segments

About Creating and Drawing Retained Graphics

An application draws by calling graphics functions. Applications store retained graphics in *segments* , which can be edited. This means that the retained image can be modified without having to re-create the unmodified portion using multiple GPI functions.

When using nonretained graphics, the output appears on the output device (for example, a window) immediately. However, if part of the picture is erased or must be repeated, the application must call the same graphics functions a second, or even a third time.

There are many other advantages to using retained graphics, including:

- **Convenience**-An application can draw retained graphics with a single function that accesses the storage area containing all the individual functions necessary for the object.
- **Flexibility**-An application can redraw graphics retained in the segment store to any number of device contexts.
- **Editing**-An application can modify an object without having to call all the functions necessary to re-create the picture. Individual segments can be moved, scaled, or rotated; then, redrawn.

 Editing the graphics within a segment is described in Editing Retained Graphics and Graphics Segments.
- **Power**-An application can access the more powerful and useful graphics functions only when graphics are stored in segments.

- **Organization**-An application can use segments to store the components that will be assembled into the final picture.

Primarily, a graphics segment is a means of grouping and storing graphics primitives and their attributes. Although the graphics within a segment usually are related in some way, they do not have to be. A segment might contain a number of unrelated GPI functions that you want to keep and execute at specific times.

Retained graphics do affect application performance, however, in that a normal presentation space requires 114KB of main memory more than a micro presentation space, and using the drawing mode DM_RETAIN adds an additional 46KB.

Note: Your application can have up to 16KB (16378) segments in the segment store of a single presentation space. The size of an individual segment is limited only by the amount of storage available to you.

Do not confuse a graphics segment with a memory segment.

Drawing Modes

When you create a presentation space, the drawing mode is set to draw. Your application can change this mode using GpiSetDrawingMode. The two additional drawing modes provided by the PM are *retain* (DM_RETAIN) and *draw-and-retain* (DM_DRAWANDRETAIN). Your application can determine which drawing mode is set by using GpiQueryDrawingMode.

The drawing mode that you select becomes current for the presentation space, and it can be changed any number of times. Select the appropriate drawing mode before creating a segment. The drawing mode affects the segment type.

Draw Mode

In *draw mode* , graphics output is provided immediately to the currently associated device and is not retained in a segment store. When the graphics have been drawn, they cannot be used again unless they are re-created. For example, when a window containing draw mode graphics is moved or sized, the graphics have to be re-created by the application.

Draw mode graphics are suitable if an application is creating fairly simple graphics quickly or is maintaining its own graphics database. In the latter case, there is nothing to gain by retaining graphics.

A segment created when the drawing mode is DM_DRAW, is called a *nonretained segment* . While it might sound contradictory, nonretained segments have certain advantages that are described in Nonretained Graphic Segments.

Retain Mode

In *retain mode* , graphics are retained in the segment store only. They are not directed to the current device as they are created.

In retain mode, the presentation space does not have to be associated with a device context when graphics are being defined. It is possible to define graphics and store their definitions without sending them to a display screen or printer. The concept of *attribute currentness* , however, is relevant only when you are drawing graphics to an output device. For example, the color or other attribute that is current when you define a primitive is the color in which the line is drawn. That color or other attribute might not be the color that is current when you you actually draw the primitive. This is described in Attribute Currentness.

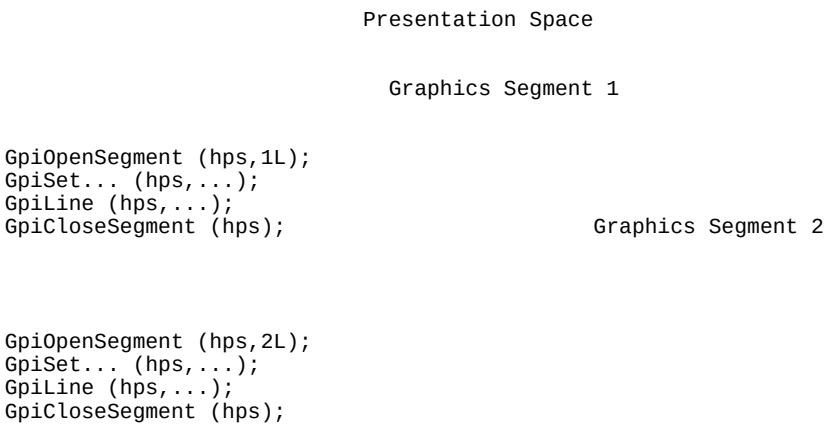
Many of the GPI queries that return current attribute values are invalid in retain mode because current attribute values have no effect when graphics are not sent to an output device.

Draw-and-Retain Mode

In *draw-and-retain mode* , graphics are both drawn on the current device as they are created and stored for later use. The GPI queries that return current attribute values are valid in draw-and-retain mode.

Creating a Graphics Segment

Your application signals the start of a graphics segment using GpiOpenSegment, which is valid only in a normal presentation space. The following figure is an example of two segments in a presentation space.



Graphics Segments in a Presentation Space

GpiOpenSegment accepts, as input, the presentation space handle and a long integer value, which names each segment.

OS/2 applications identify segments with long integer values. If you want to refer to the segment individually in later graphics operations (for example, to edit the segment) the identifier you supply must be greater than 0 and unique within the presentation space.

If you do not want to refer to the segment individually after it is created, you can give the segment an identifier of 0. You might do this when you are creating nonretained segments, for example. Any number of segments can have the 0 identifier. They are referred to throughout this book as *zero segments* .

To determine which names already have been used in a presentation space, use GpiQuerySegmentNames. This function returns an array of segment names for all retained segments, excluding zero segments, within a specified name range.

Segments do not have to be named in consecutive order (although it is recommended for organizational purposes) because the segment order can be changed using GpiSetSegmentPriority.

Filling a Graphics Segment

After a graphics segment is opened, the GPI functions that follow become a part of a retained segment and are executed every time the segment itself is drawn.

The typical GPI functions that would be called are those that:

- Create graphics primitives-such as lines or markers
- Assign attributes to those primitives-such as color or line type.

However, there are a few GPI functions that you cannot call while there is an open segment-for example a second GpiOpenSegment. These functions are identified in GPI Function Context.

A presentation space can contain many segments. Each time you open a new segment, many attributes reset themselves to default values. Therefore, the current position and attribute values that apply before you call GpiOpenSegment cannot be guaranteed to be in effect after you call the function. Beginning each segment with a number of attribute-setting requests and, possibly, with GpiSetCurrentPosition, is recommended. Your application also might take advantage of GpiSetDefAttrs, for example, to minimize the number of attributes that must be dealt with upon opening a new segment.

Closing a Graphics Segment

When you have finished creating a graphics segment, close it using GpiCloseSegment. There can be only one open segment at a time in a single graphics presentation space, so you must close one segment before going on to the next.

There is some degree of clean-up processing associated with using GpiCloseSegment, known as *close-segment processing* , that can make current attribute values unreliable. For more information about the effects of GpiOpenSegment and GpiCloseSegment on current attributes, see Graphics Attributes.

Segment Attributes

Each segment, whether retained or nonretained, has a number of characteristics, called *attributes* that you can set in accordance with your application's requirements. The attributes of a segment are quite different from those of a graphics primitive in that segment attributes have ON and OFF settings. There are seven segment attributes. However, only two are described in this chapter: *chained* (ATTR_CHAINED) and *fast-chained* (ATTR_FASTCHAIN).

When an application creates a segment in a presentation space, the operating system assigns initial attributes to it. By default, five of the attributes will be set ON and two will be set OFF. The chained and fast-chained attributes are set ON. Your application can alter these values using GpiSetInitialSegmentAttrs or retrieve the values of the current initial attributes using GpiQueryInitialSegmentAttrs.

The chain attribute tells the operating system to add each new segment in your application's presentation space to the segment chain. The fast-chained attribute tells the operating system to prevent the resetting of the primitive attributes to their default values before drawing the segment chain.

Chained Attribute

When you define a segment with the chained attribute switched ON, that segment becomes a part of the *segment chain* and is called a *chained segment* . The segment chain is composed of all chained segments defined in a single presentation space. Segments can be chained together so that they can be drawn as a group. By default, each new segment is chained to the previous segment.

There can be only one segment chain at a time in a single presentation space, and all chained segments are chained to each other in the order in which you created them. Zero segments must have the chained attribute.

Usually, a logical relationship exists between the segments in a segment chain, although this is not a requirement. The whole segment chain can be drawn using GpiDrawChain. Each segment in the chain is called a *root segment* . Root segments are affected by those GPI functions that act on the segment chain, but they also can be manipulated independently of the chain.

Segments that are defined with the chained attribute switched OFF are called *unchained segments* . Your application can switch off the chain attribute using GpiSetInitialSegmentAttrs (hps, ATTR_CHAIN, ATTR_OFF). Unchained segments always are retained when they are created, regardless of the current drawing-mode parameter. An unchained segment must have a unique name.

There are a number of reasons for defining a segment as unchained. For example, a particular segment might not belong to the picture that

is defined by the segment chain. You also are likely to define as unchained any segment that belongs to the picture but that has no single, fixed place in the segment chain. A car wheel, for example, could be defined once but drawn four times in a picture of a car. It would have no single, fixed place in the segment chain, but would be included four times in the picture by being called from one or more root segments.

A segment is called from another segment by including GpiCallSegmentMatrix in the calling segment. A segment and the segments it calls logically are one object. An important point about called segments is that they assume the primitive attribute settings of the calling segment. Of course, you can change the attribute settings within the called segment if the inherited values are inappropriate.

A closed, unchained segment can be called from any other segment. Chained segments, however, cannot be called from other segments.

Fast-Chained Attribute

The *fast-chained attribute* applies only to chained segments. It prevents primitive attributes from being reset to their default values at the beginning of the segment, an aid to performance. There is unnecessary overhead in resetting attributes to their defaults if either of the following is true:

- You are going to change the default values of the attributes at the start of the segment.
- You know that attributes have not been altered previously from their default values.

The fast-chained attribute is switched ON by default, and you should leave it on unless you specifically want attributes to be set to their default values. To turn off the fast-chained attribute, call GpiSetInitialSegmentAttrs (hps, ATTR_FASTCHAIN, ATTR_OFF).

Actual Drawing Mode

The drawing mode, as defined by GpiSetDrawingMode, works in conjunction with the segment status to produce the *actual drawing mode* . It is the actual drawing mode that determines whether graphics are:

- Drawn directly to the output device
- Retained in the segment store
- Both drawn and retained

The actual drawing mode is summarized in the following table.

The Current Drawing Mode

		Context	
GpiSetDrawingMode parameter	Chained Segment	Unchained Segment	Outside of any segment
DM_DRAWANDRETAIN	draw-and-retain	retain	draw
DM_RETAIN	retain	retain	draw
DM_DRAW	draw	retain	draw

As you can see in the preceding table, graphics within chained segments always conform to the current drawing mode parameter. That is, they are drawn in DM_DRAW mode, retained in DM_RETAIN mode, and both drawn and retained in DM_DRAWANDRETAIN mode.

Graphics in unchained segments always are retained, regardless of the current drawing-mode parameter. You cannot retain segments created when the current drawing-mode parameter is DM_DRAW, unless they are unchained segments.

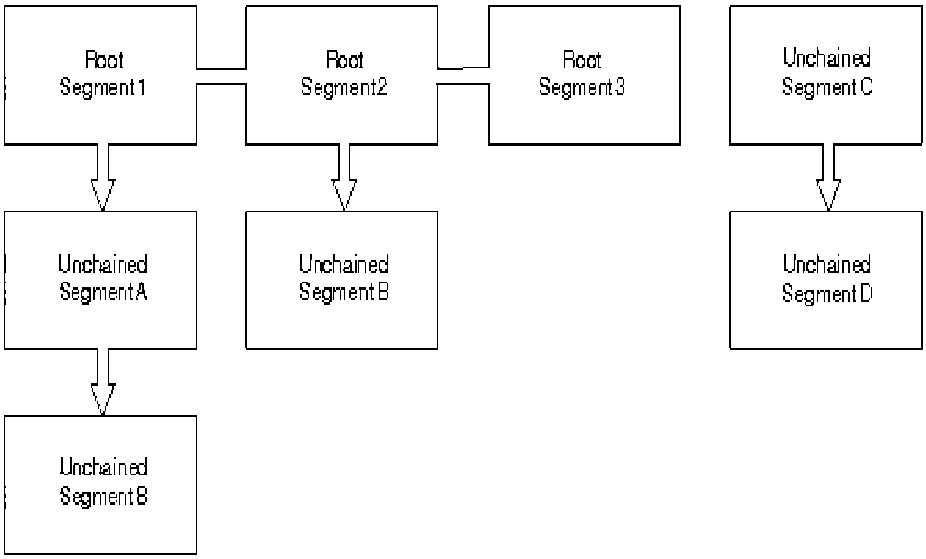
Similarly, graphics outside segments always are drawn without being retained. You cannot retain primitives outside segments, regardless of the current drawing mode.

Note: A micro presentation space has no segment store. Therefore, you cannot create graphics segments in a micro presentation space, nor can you change the drawing mode, which is always DM_DRAW.

Drawing a Retained Graphic

Your application can draw a complete segment chain with GpiDrawChain. The segments are drawn in the order in which they appear in the chain. For example, the segments in the following figure are drawn in the following order:

1. Root segment 1
2. Unchained segment A
3. Unchained segment B
4. Root segment 2
5. Unchained segment B
6. Root segment 3.



Chained and Called Segments

Segment A is called by root segment 1, and segment B is called by both segment A and root segment 2. Segment C calls segment D. Both C and D are unchained segments that are not called from the segment chain, and consequently, are not part of the segment chain.

GpiSetSegmentPriority changes the position of a root segment (which must not be a zero segment) in the chain. As input to this function, you supply the name of the segment you want to reorder and the name of a reference segment. The *reference segment* is the segment that either will be immediately before, or immediately after, the reordered segment's new position in the chain.

If the segment you are moving is to come after the reference segment in the chain, it is said to have a *higher priority* (HIGHER_PRI). If it is to come before the reference segment, it is said to have a *lower priority* (LOWER_PRI). The nearer a segment is to the end of the chain, the higher its priority.

If you supply the name of an unchained segment as input to GpiSetSegmentPriority, the segment is added to the chain in the position you specify. To learn the position of a segment in the chain, use GpiQuerySegmentPriority.

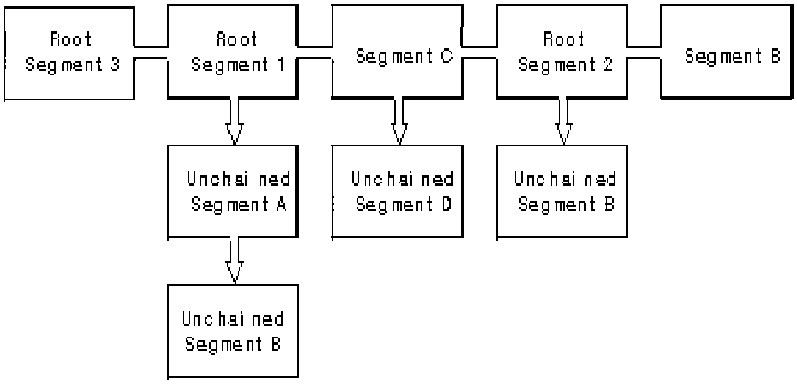
If your application called the following functions:

```
GpiSetSegmentPriority (hps, C, 2, LOWER_PRI)
GpiSetSegmentPriority (hps, 3, 0, HIGHER_PRI)
GpiSetSegmentPriority (hps, B, 0, LOWER_PRI)
```

the segment chain in the previous figure would appear as in the following figure; and GpiDrawChain would draw the segments in the following order:

1. Root segment 3
2. Root segment 1
3. Unchained segment A
4. Unchained segment B

5. Segment C
6. Unchained segment D.
7. Root segment 2
8. Unchained segment B.
9. Segment B



Chained and Called Segments Reordered Using GpiSetSegmentPriority

Segment Priority

The priority of a segment, in conjunction with the current foreground and background mix attributes, affects how the picture is presented to the user. The segment with the highest priority is drawn last, and appears on top of the previously drawn primitives. Picture components in segments with low priorities risk being drawn over and never seen by the user. The mix attributes affect the graphics functions within each segment.

GpiDrawSegment

GpiDrawSegment accepts as input any segment name greater than zero. GpiDrawSegment can draw individual segments both inside and outside the segment chain. Any segment that is called from the GpiDrawSegment-named segment also is drawn.

GpiDrawFrom

You can draw a smaller portion of the entire segment chain (but a larger portion than a single segment) using GpiDrawFrom. GpiDrawFrom, as input, a presentation space handle and two nonzero segment names, the first of which must be part of the segment chain. GpiDrawFrom then draws the segments, starting with the first and ending with the last, including all chained and called segments.

Attribute Modes

You can change primitive-attribute settings at any time. The new values you specify replace the existing values. For example, if the current foreground color is green and you change it to red, red replaces green as the foreground color attribute.

When you set primitive attributes within a retained or nonretained segment, you can save the existing attribute values on a last-in-first-out (LIFO) stack, from where they can later be retrieved and made current again. You do this by selecting either of the two attribute modes:

- AM_PRESERVE mode-Also known as *push-and-set mode*
- AM_NOPRESERVE mode-Also known as *set mode* .

In AM_NOPRESERVE mode, which is the default setting, existing attribute values are replaced by any new attribute values that you supply. In AM_PRESERVE mode, the existing attribute values are stored on a LIFO stack, and then the new values that you specify take effect. AM_PRESERVE mode also causes the current position to be saved if the position was specified using GpiSetCurrentPosition. If you use GpiMove to set the current position, the position is not saved, regardless of the attribute mode.

To select a current attribute mode, use GpiSetAttrMode. The current attribute mode affects subsequent attribute-setting requests in the presentation space. Attribute modes are not applicable to micro presentation spaces, because graphics segments can be created only in normal presentation spaces.

To retrieve an attribute value from the LIFO stack, use GpiPop, which pulls back the attribute that was stored most recently on the stack. You can retrieve more than one attribute value by supplying a number as an input parameter of this function. For example, if you specify 4, GpiPop retrieves the four attributes that were most recently stored on the stack. If each of the values retrieved applies the same type of attribute to the same primitive (for example, if all four are line-type settings), the last one to be retrieved (and, therefore, the first one on the stack) becomes the current setting.

If you save attribute values from any segment called from another segment, and do not retrieve the values using GpiPop, the values are restored automatically when the end of the segment is reached. If you save attribute values from any segment that is not called from another segment and do not explicitly restore those values using GpiPop, they are lost at the end of the segment.

The AM_PRESERVE mode is useful when you do not want attributes in calling segments to be overwritten by attributes specified in the called segments. This overwriting happens because a calling segment and the segments it calls are logically one object. Attribute changes within a called segment remain current upon return to the calling segment. If you set some attribute values at the start of a called segment, and the current attribute mode is AM_PRESERVE, the attribute values of the calling segment are stored on the LIFO stack. At the end of the called segment, the values on the stack are retrieved automatically so that the calling segment continues with its own attribute values.

Reusing the Presentation Space

A normal presentation space with segments retained in the segment store can be costly in terms of storage. Therefore, delete any presentation space that you no longer need, using GpiDestroyPS. If, for example, your application is using a series of presentation spaces, each with a different format, the creation and deletion activity also can be costly. The PM provides the following functions that let you reuse or redefine an existing presentation space so you can create a presentation space once and use it any number of times.

- GpiSavePS
- GpiRestorePS
- GpiResetPS
- GpiSetPS

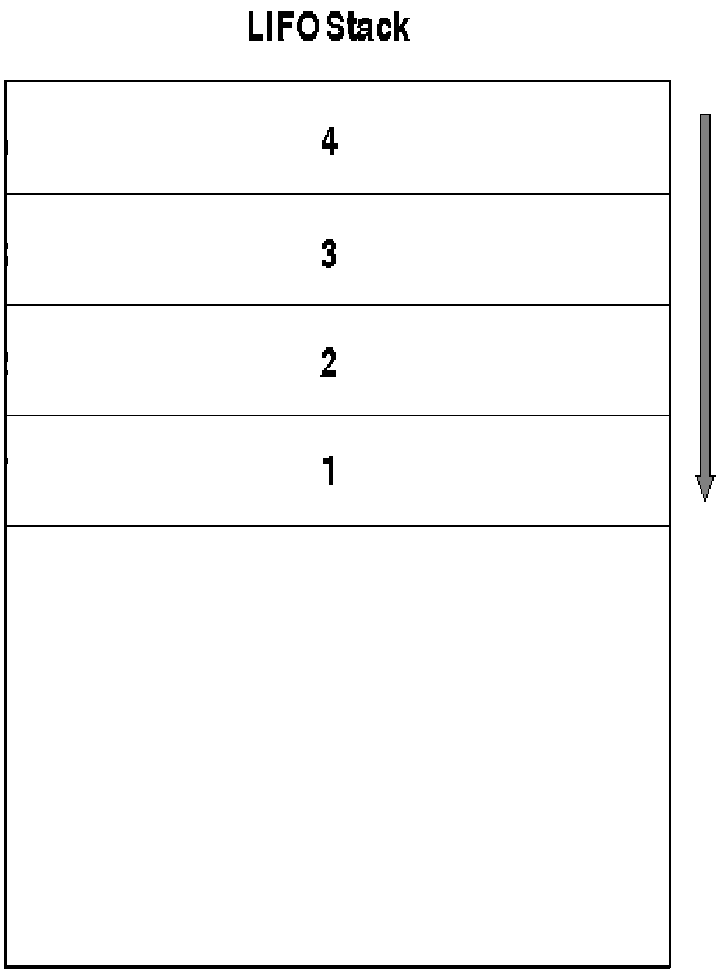
GpiSavePS

GpiSavePS saves presentation space information (such as current primitive attributes and the current position) on a dedicated LIFO stack. The presentation space itself is unchanged; that is, it still exists, has the same presentation space handle, and the presentation page dimensions and format are the same.

Output from GpiSavePS is a number that identifies the saved information on the LIFO stack. An output of 3, for example, tells you that this is the third lot of presentation space data on the stack.

GpiRestorePS

GpiRestorePS retrieves the saved information from the LIFO stack and reapplies it to the presentation space. Input to this function can be either the identifier returned to you from GpiSavePS or a relative value. A relative value of -1, for example, retrieves the information most recently stored on the stack. You do not have to restore the information that was most recently saved. However, any data that you skip over is discarded.



A LIFO Stack with Four Items

GpiResetPS

When a presentation space is first created, it is in a neutral state. Current attributes are set to their initial default values. The current position is (0,0). The segment store contains no segments, and there are no application-defined resources, such as logical color tables.

You can return an existing presentation space to this state using GpiResetPS. GpiResetPS has three levels of reset activity, each more powerful than the last. These are:

1. GRES_ATTRS, which is equivalent to the processing done by GpiCloseSegment.
2. GRES_SEGMENTS, which is equivalent to creating a new presentation space, but without deleting any logical resources. All retained segments are deleted from the segment store.
3. GRES_ALL, which is equivalent to creating a new presentation space.

GpiResetPS does not alter the size or format of the presentation page.

GpiSetPS

GpiSetPS redefines the size and format of the presentation page. The processing performed when you call GpiSetPS is similar to that performed by GpiCreatePS, except that the presentation space already exists. The presentation space is returned to a neutral state (which is the equivalent of requesting GRES_ALL using GpiResetPS), and the presentation page is redefined. For example, you can change a presentation page defined in 0.01mm units to one defined in pels. Essentially, this lets you work with a new presentation space without having to delete one and create another.

You also can use GpiSetPS to change the current mapping mode of a presentation space. To do this, use the PS_NORESET flag, which is the equivalent of requesting GRES_SEGMENTS using GpiResetPS. This feature is particularly useful if you are designing an application that deals with page layout and drafting, or if you want the screen size to correspond to the page size for the printed output.

Nonretained Graphic Segments

It is valid segment construction to create a segment bracket while the drawing mode is DM_DRAW. The GPI functions contained within the bracket are drawn immediately rather than being retained for future use. This type of segment is called a *nonretained segment* .

Because the usual reason for constructing segments is to take advantage of retained graphics, nonretained segments might appear as a contradiction in terms at first. However, there are four particular advantages in their use:

- If ATTR_FASTCHAIN is set to OFF, the GpiOpenSegment implicitly resets all attributes to their defaults.
- The GpiOpenSegment and GpiCloseSegment initialize and reset the viewing transform matrix, just as they do for retained segments.
- Nonretained segments can be recorded in a metafile, just as a retained segment can.
- A graphic in a nonretained segment, with a unique name, can be converted easily to a retained graphic for future use.

Using Segment Creating and Drawing Functions

You can use retained-drawing and segment functions to:

- Create a chained or called segment
- Draw the picture associated with one or more segments

Creating a Chained Segment

To create a chained segment, you must:

1. Set the drawing mode to DM_RETAIN.
2. Check to see if the chained attribute is one of the initial segment attributes using GpiQueryInitialSegmentAttrs.
3. Set the chained attribute, if necessary, with GpiSetInitialSegmentAttrs.

4. Open the segment using GpiOpenSegment.
5. Perform the necessary drawing operations.
6. Close the segment using GpiCloseSegment.

The following figure is an example of a segment containing a box primitive and calling another segment using GpiCallSegmentMatrix.

```
#define INCL_GPISEGMENTS
#define INCL_GPITRANSFORMS
#include <os2.h>
void fncSEGS01(void){
    POINTL ptl;
    HPS hps;
    LONG idSegment    = 1;
    LONG idNonChained = 2;
    MATRIXLF matlfTransform = { MAKEFIXED(2,0), MAKEFIXED(0,0), 0,
                                MAKEFIXED(0,0), MAKEFIXED(1,0), 0,
                                0, 0, 1 };

    /*********************************************************************/
    /* Turns chaining on. Adds the new segment to the segment chain.     */
    /* Segment idNonChained is called, whether chained or not.           */
    /*********************************************************************/

    if (ATTR_OFF == GpiQueryInitialSegmentAttrs(hps, ATTR_CHAINED)
        GpiSetInitialSegmentAttrs(hps, ATTR_CHAINED, ATTR_ON);
    GpiOpenSegment(hps, idSegment);
    ptl.x = 150; ptl.y = 150;
    GpiMove(hps, &ptl);
    ptl.x = 225; ptl.y = 225;
    GpiBox(hps, DRO_FILL, &ptl, 0L, 0L);
    GpiCallSegmentMatrix(hps, idNonChained, 9L, &matlfTransform,
        TRANSFORM_REPLACE);
    GpiCloseSegment(hps);
} /* fncSEGS01 */
```

Creating a Called Segment

To create a called segment, you must:

1. Set the drawing mode to DM_RETAIN.
2. Check to see whether the chained attribute is one of the initial segment attributes using GpiQueryInitialSegmentAttrs.
3. Set the chained attribute, if necessary, with GpiSetInitialSegmentAttrs.
4. Open the segment using GpiOpenSegment.
5. Perform the necessary drawing operations.
6. Close the segment using GpiCloseSegment.

The following figure shows an example of how to draw a box in a called segment.

```
#define INCL_GPISEGMENTS
#define INCL_GPICONTROL
#include <os2.h>
void fncSEGS02(void){
    POINTL ptl;
    HPS hps;
    LONG idNonChained = 2;

    GpiSetDrawingMode(hps, DM_RETAIN);

    /* Creates a non-chained segment. */
```

```
    if (ATTR_ON == GpiQueryInitialSegmentAttrs(hps, ATTR_CHAINED))
        GpiSetInitialSegmentAttrs(hps, ATTR_CHAINED, ATTR_OFF);
    GpiOpenSegment(hps, idNonChained);
    ptl.x = 100;
    ptl.y = 100;
    GpiMove(hps, &ptl);
    ptl.x = 200;
    ptl.y = 200;
    GpiLine(hps, &ptl);
    GpiCloseSegment(hps);
} /* fncSEGS02 */
```

Drawing a Segment Chain

The following figure shows an example of how to draw a segment chain using GpiDrawChain.

```
#define INCL_GPICONTROL
#include <os2.h>
void fncSEGS03(void){
    HPS hps;

    if (DM_DRAW != GpiQueryDrawingMode(hps))
        GpiSetDrawingMode(hps, DM_DRAW);
    GpiDrawChain(hps);
} /* fncSEGS03 */
```

Character String Primitives

Character string primitives are printed or displayed text limited to a length of 256 characters by the PM.

The following topics are related to information in this chapter:

- Presentation spaces and device contexts
- Color and mix attributes
- Fonts
- Coordinate spaces and transformations

About Character String Primitives

A PM application must make certain choices about fonts in preparation for text display or printing. Often these choices are driven by selections from a user. After the selections are made, as discussed in Fonts, the application can make additional decisions about the appearance of the individual characters within the font by setting attributes in the CHARBUNDLE data structure. CHARBUNDLE is the lowest of three levels of data structures that define the appearance of displayed or printed text. The other two, FONTMETRICS and FATTRS, are described in Fonts.

A font family-for instance, Helvetica** or Courier-is a collection of fonts. Fonts within the same family share certain attributes such as *stroke width* and *serif characteristics* . Stroke width refers to the width of lines used to draw characters and symbols from a font. A serif is a short crossline drawn at the ends of the main strokes that form a character or symbol.

Individual fonts within a family differ from each other in the following ways:

- Height
- Line weight
- Appearance

Height refers to the point size of a font. A common example of line weight is **boldface**. A common example of appearance is *italic* .

The most important factor that affects the CHARBUNDLE attributes is whether the current font is an *image* or an *outline* font. The following figure shows an example of the difference.

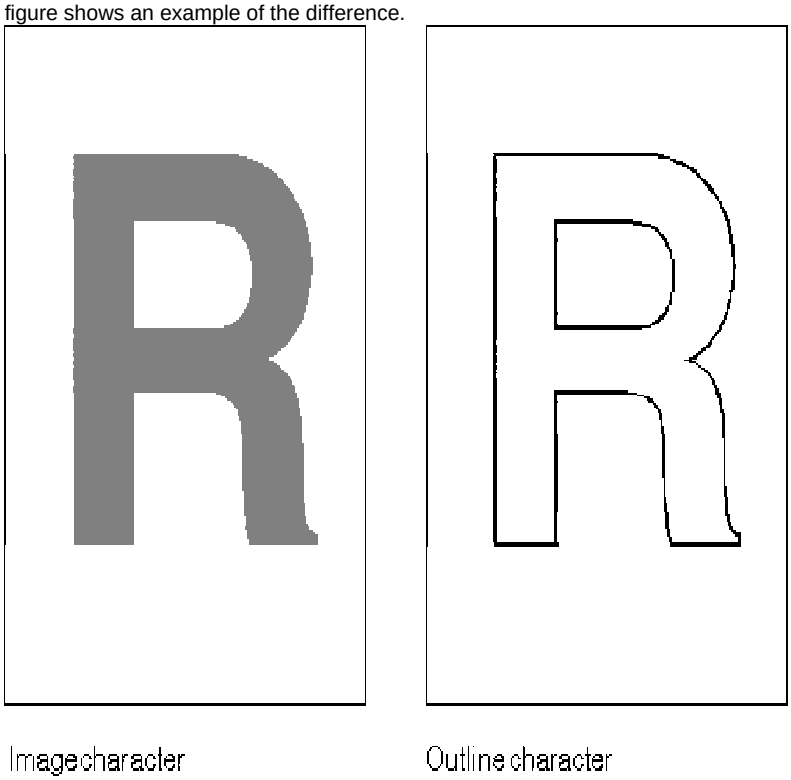


Image and Outline Fonts

Outline fonts are composed of characters drawn with straight and curved lines. Image fonts, also called bit map fonts, are composed of pels arranged in certain shapes.

Attributes of Character String Primitives

Attributes of the character string primitives contained in CHARBUNDLE are as follows:

- Character set
- Character mode
- Character cell
- Character angle
- Character shear
- Character direction
- Character text alignment
- Character extra
- Character break extra
- Foreground color
- Background color
- Foreground mix
- Background mix

When an application creates a presentation space, the character string attributes are set to the default values shown in the following table.

Character String Attribute Default Values

Attribute	Default Value	Function that Redefines Attribute
Set	LCID_DEFAULT	GpiSetCharSet
Mode	CM_MODE1	GpiSetCharMode
Cell	Outline font - Defined by device Image font - Defined by font	GpiSetCharBox
Angle	(1,0)	GpiSetCharAngle
Shear	(0,1)	GpiSetCharShear
Direction	Left to right	GpiSetCharDirection
Text alignment	Left	GpiSetTextAlignment
Extra	0	GpiSetCharExtra
Break extra	0	GpiSetCharBreakExtra
Foreground color	Black	GpiSetAttrs(CBB_COLOR)
Background color	Clear	GpiSetAttrs(CBB_BACK_COLOR)
Foreground mix	Overpaint	GpiSetAttrs(CBB_MIX_MODE)
Background mix	Leave alone	GpiSetAttrs(CBB_BACK_MIX_MODE)

Current character attributes take effect when you send graphics characters to an output device. They have no effect at the time you define a logical font.

Character Set

Character set defines the local identifier, *lcid*, of the current font. The font lcid is set using GpiSetCharSet. GpiCreateLogFont associates the font with an lcid.

The value of the current lcid is determined using GpiQueryCharSet.

Character Mode

Every presentation space has a current *character mode* , which determines the extent to which a font can be affected by the attributes defined in CHARBUNDLE. The mode affects compatibility issues that are invisible to both the user and the programmer. The following table describes the character modes and their influences on the current font.

An application selects a character mode using GpiSetCharMode. When a mode is selected, it applies to any font (including the system font) used while the mode is current. The character mode takes effect when you draw character strings; it has no effect at the time you define a logical font.

Character Mode Effects on Character Attributes

Character Mode	If Image Font...	If Outline Font...
CM_MODE1	Ignores all current character attributes, except character direction.	Affected by all current character attributes.
CM_MODE2	Uses the character shear, box, angle, and direction attributes to position the character cell, but the characters themselves are not sheared, rotated, scaled, or reversed.	Affected by all current character attributes.
CM_MODE3	Raises an error if you try to draw a character string. Note: Any font used when the current character mode is CM_MODE3 *must* be an outline font.	Affected by all current character attributes.

The default character mode (CM_DEFAULT) is identical to CM_MODE1.

Character Cell

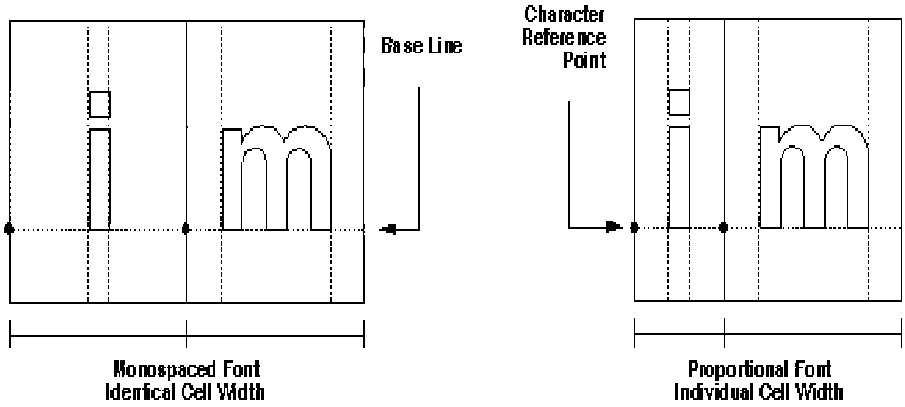
A *character cell* is an imaginary rectangular boundary that defines the horizontal and vertical space occupied by a single character from an outline character set.

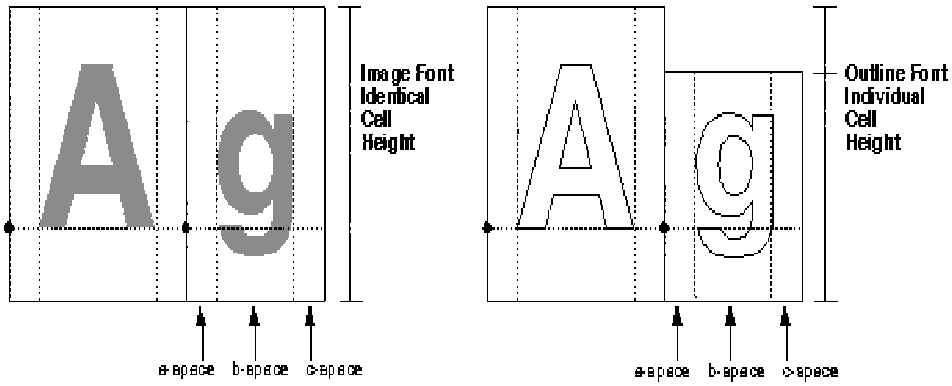
The PM calculates character cell width and height from the point size. The width value for the character cell is the nominal width of the lowercase characters in the character set. In a *monospace* font, the width of all character cells is identical. In a *proportional* font, the width of the character cells depends on the character.

In an image font, the height of the character cell is the number of pels in the font. In an outline font, the height of the character cell is the point size of the font.

The characters in a character string are positioned one character per cell. The spacing between adjacent characters in a string is affected by the character cell attribute, except for image characters in CM_MODE1.

Cell width determines the spacing of consecutive characters along the baseline, as illustrated in the following figure.

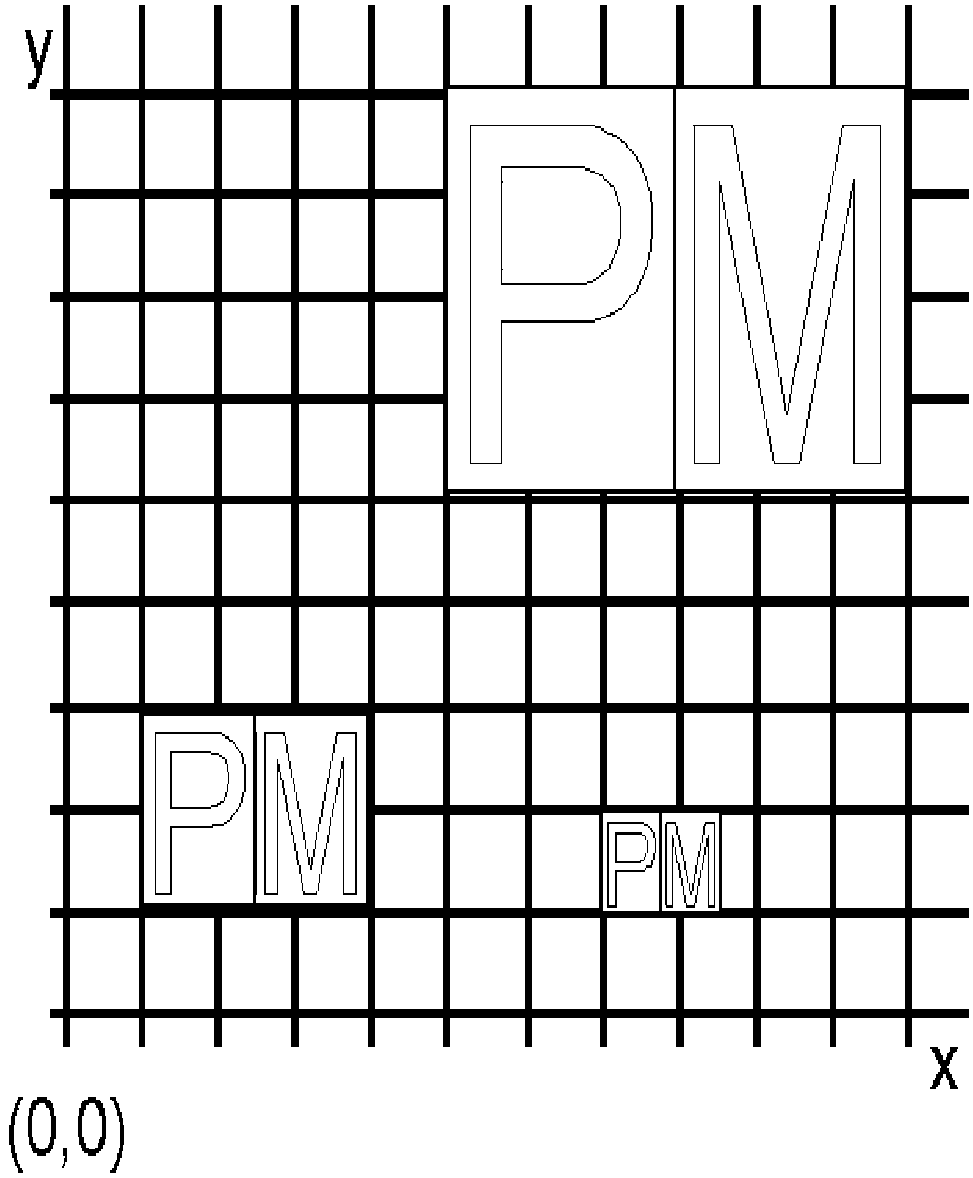




Character Cell Measurements

Current cell size is specified using GpiSetCharBox. As input, this function accepts the desired height and width of the character cell in world coordinates. These values are related to certain dimensions in the FONTMETRICS structure that controls font attributes. Heights or widths of 0 are valid input and cause the outline character to be drawn as a point or straight line. Heights or widths of negative values cause certain special effects, for example, reversed lettering.

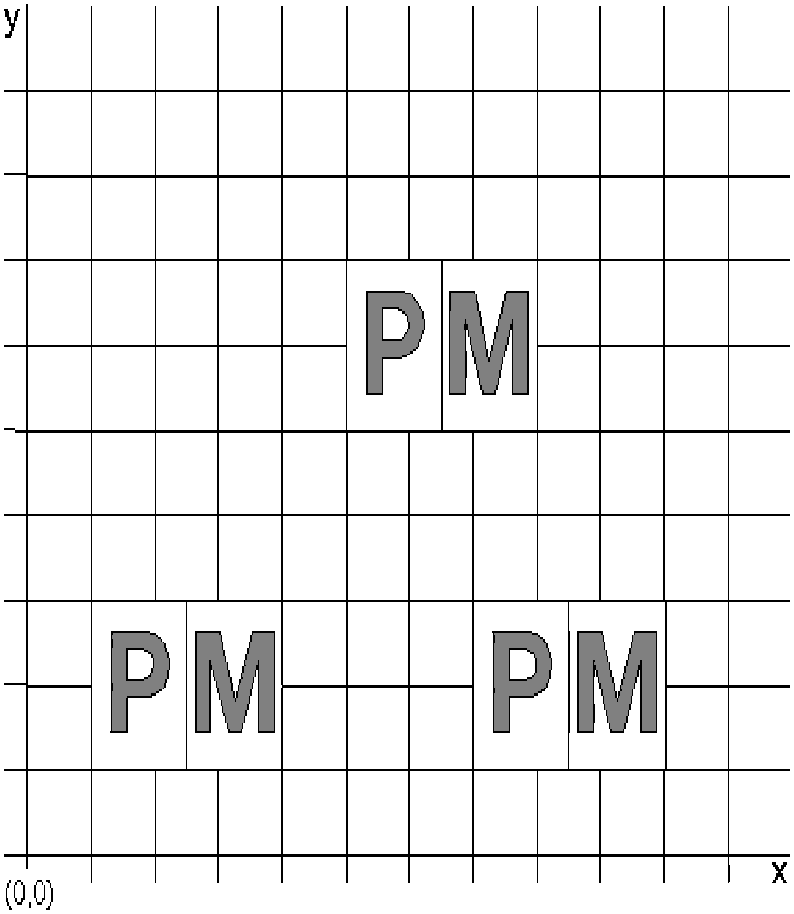
The character cell value affects both the size and position of characters drawn from an outline font, regardless of the current character mode. Each character is scaled up or down to fit the cell size, as shown in the following figure.



Effect of the Character Cell on an Outline Font

The character cell value is ignored if the current font is an image font and the current character mode is CM_MODE1, as shown in the following figure.

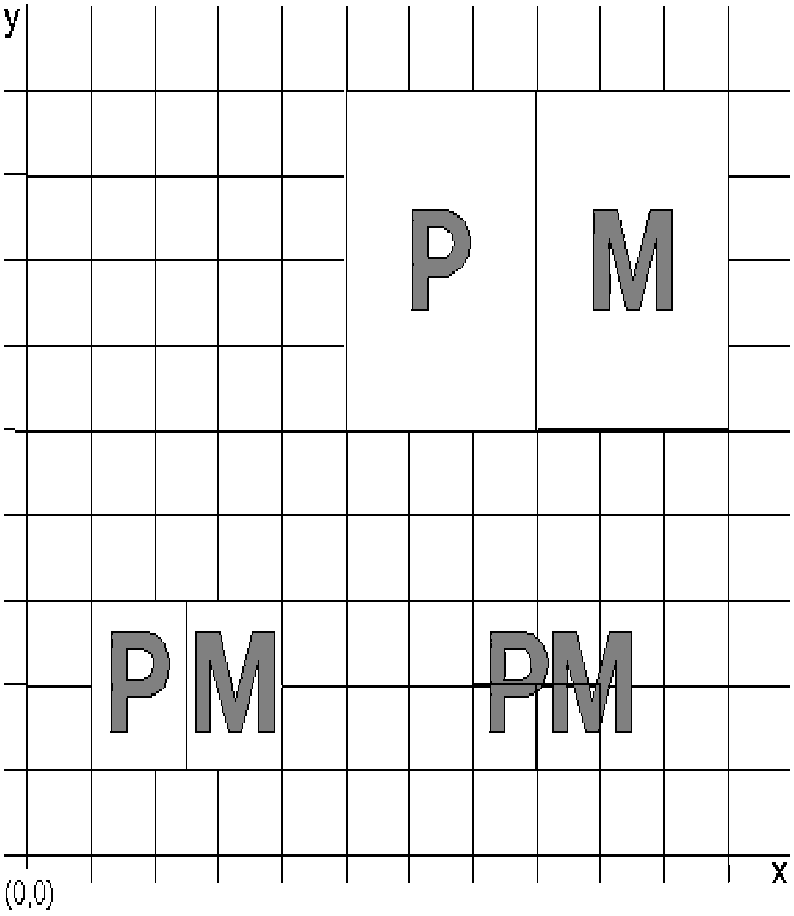
Note: It is essential to code the character cell correctly, even if you anticipate using image fonts. In case of a font match failure, an outline font can be substituted for a image font.



Effect of the Character Cell on an Image Font in CM_MODE1

Although the character cell has been both increased and decreased, the character string is unaltered.

The character cell value controls the *positioning* of image-font characters when the current character mode is CM_MODE2; but it cannot cause the characters to be scaled to fit the cell. This effect is shown in the following figure.



Effect of the Character Cell on an Image Font in CM_MODE2

If you increase the character cell size for an image font in CM_MODE2, the characters are more widely spaced, but their size is not changed. If you decrease character cell size, the space between the characters is reduced, and because the characters themselves cannot be scaled, they can overlap.

Default Character Cell

The default character-cell size is a device-dependent value. You do not need to know this value unless you want to switch back to the initial default value after having specified another value. It is recommended that you save initial default values using GpiSavePS.

The default character cell, like other default attributes, can be set using GpiSetDefAttrs(CBB_BOX). If necessary, you can query the current default value using GpiQueryDefAttrs.

Coding a Character Cell

The dimensions that an application passes to GpiSetCharBox and that GpiQueryCharBox returns are fixed-point values. A *fixed* value is a 32-bit value whose high-order 16 bits contain the integral part of the floating-point number and whose low-order 16 bits contain the fractional part. The fractional part is the numerator of a fraction whose denominator is fixed at 65536. The MAKEFIXED macro provides a method for producing fixed values. If, for example, one of the character cell dimensions were 7.635 world units, an application could obtain the corresponding fixed value by using the MAKEFIXED macro, passing 7 as the first argument and 41615 as the second.

The purpose of the character cell is to assist other font metrics to define text lines. For example, if you planned to have an average text line of 60 characters, dividing your output width by 60 would provide your character cell width.

DevQueryCaps can be used to provide information about suitable character cell values. DevQueryCaps returns two sets of values that can influence the character cell:

- **Default cell values**
 - CAPS_GRAPHICS_CHAR_WIDTH
 - CAPS_GRAPHICS_CHAR_HEIGHT.
- **Device resolution**
 - CAPS_HORIZONTAL_FONT_RES
 - CAPS_VERTICAL_FONT_RES.

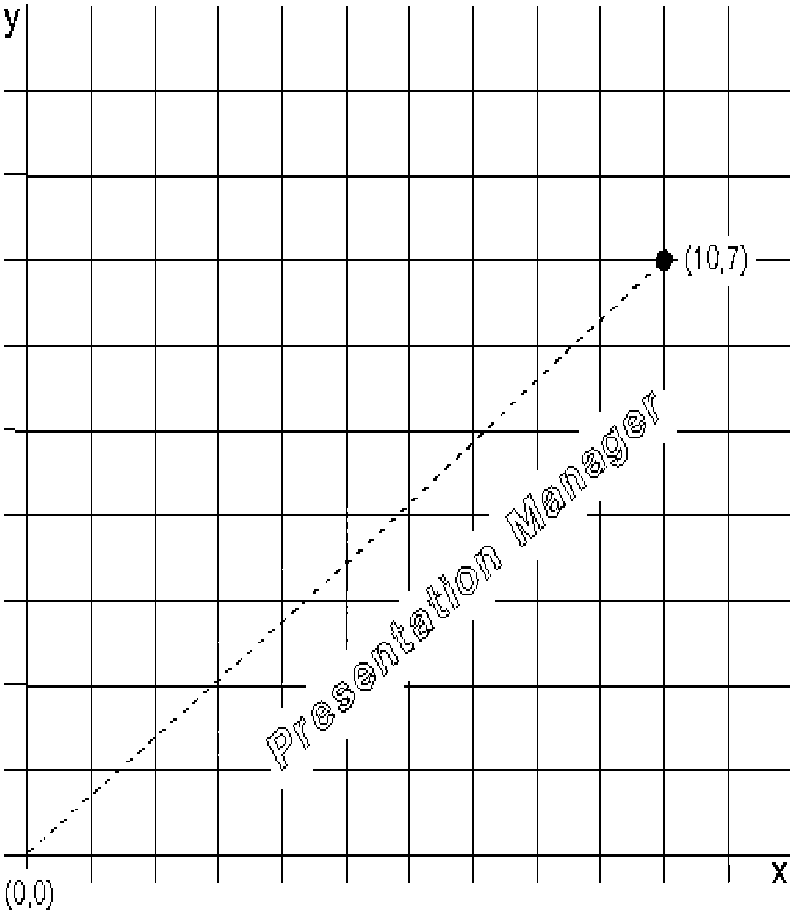
The default cell values return the size of the default character cell in pels. Convert the device resolution into character cell values by multiplying it by the desired point size, then dividing by 72 (72.2818).

Character Angle

The *character angle* is defined by the x-axis and a vector drawn through the origin to a specified point in a Cartesian coordinate system. Neither (0,0) nor the specified point need have any relation to the current position. The operating system then aligns the baseline with this vector.

An application can retrieve the point that defines the character angle vector using GpiQueryCharAngle. An application can set the character angle using GpiSetCharAngle. This function accepts as input the x- and y-coordinates of a point that defines the new vector. When you specify an angle, it becomes the current setting. To reset the character angle vector to its default value (parallel to the x-axis), call GpiSetCharAngle with both x and y equal to 0.

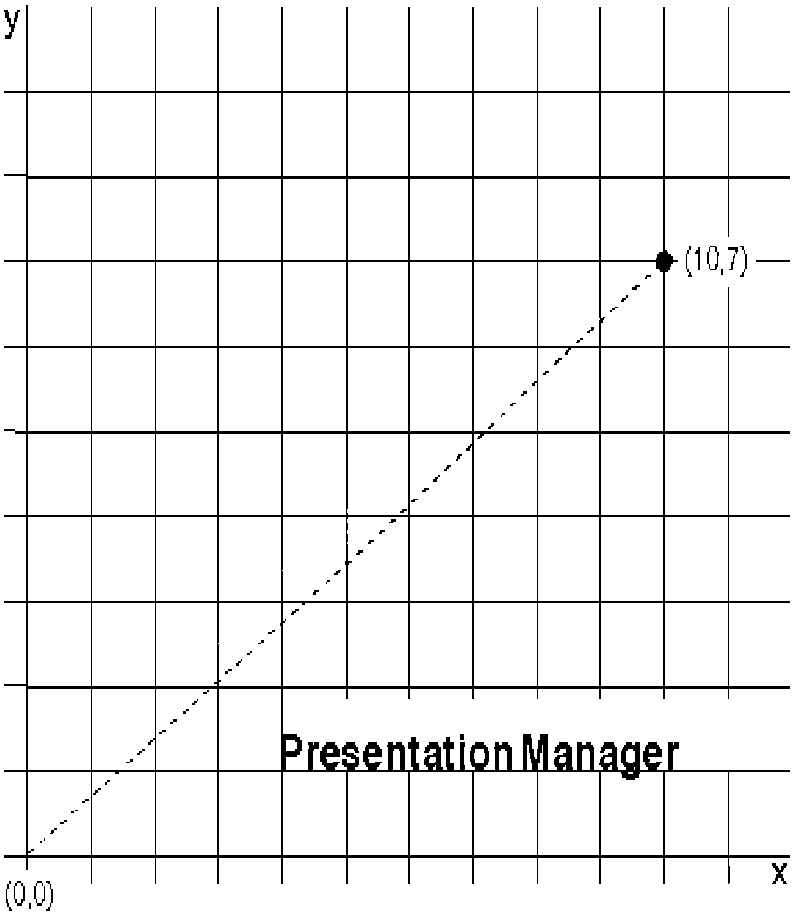
The effects of the current character angle vary depending on the current character mode and the current font type. When the current font is an outline font, the current character angle determines the angle of both the whole string and the individual characters in the string, regardless of the current character mode. The baseline of each character cell is drawn parallel to the new baseline, as shown in the following figure.



Effect of the Character Angle on an Outline Font

The character string is drawn parallel to the vector drawn from the origin to (10,7). The baseline of each of the character cells also is drawn parallel to this vector.

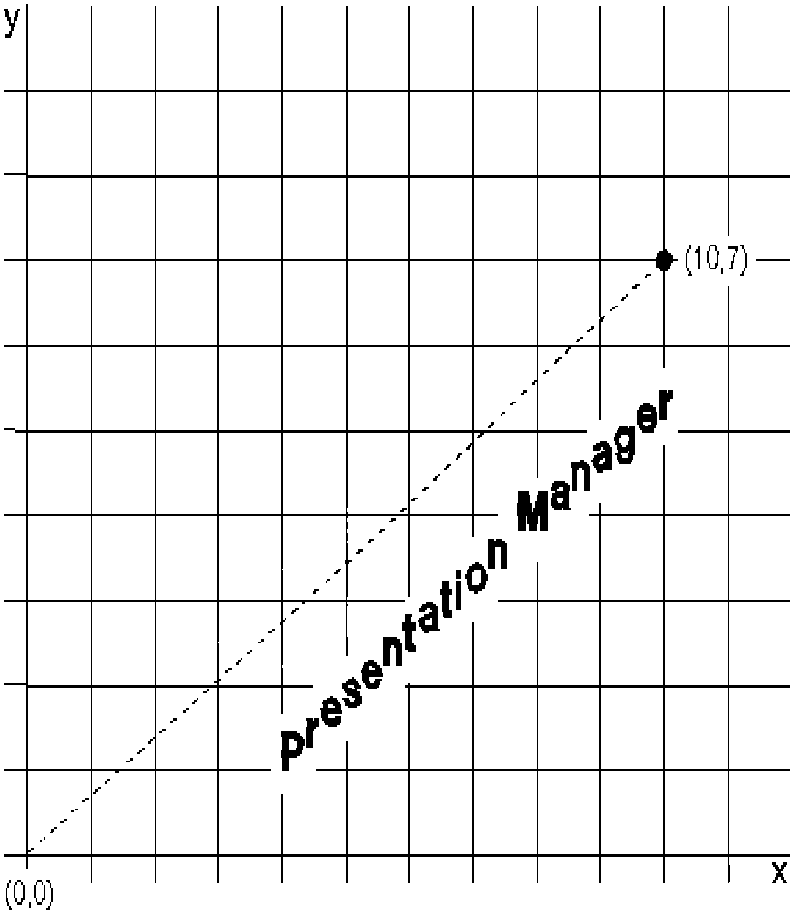
The character angle value is ignored if the current font is an image font and the current character mode is CM_MODE1, as shown in the following figure.



Effect of the Character Angle on an Image Font in CM_MODE1

The angle of the character string is unaltered by GpiSetCharAngle.

When the current font is an image font and the current character mode is CM_MODE2, the current character angle determines the angle of the whole string but does not affect the individual characters in the string. The *character reference point*, which is the point at which the character baseline intersects the left edge of the character cell, is placed on a line parallel to the new baseline. The baseline of each character cell remains parallel to the x-axis, as shown in the following figure.



Effect of the Character Angle on an Image Font in CM_MODE2

The character string is drawn parallel to the vector from (0,0) to (10,7).

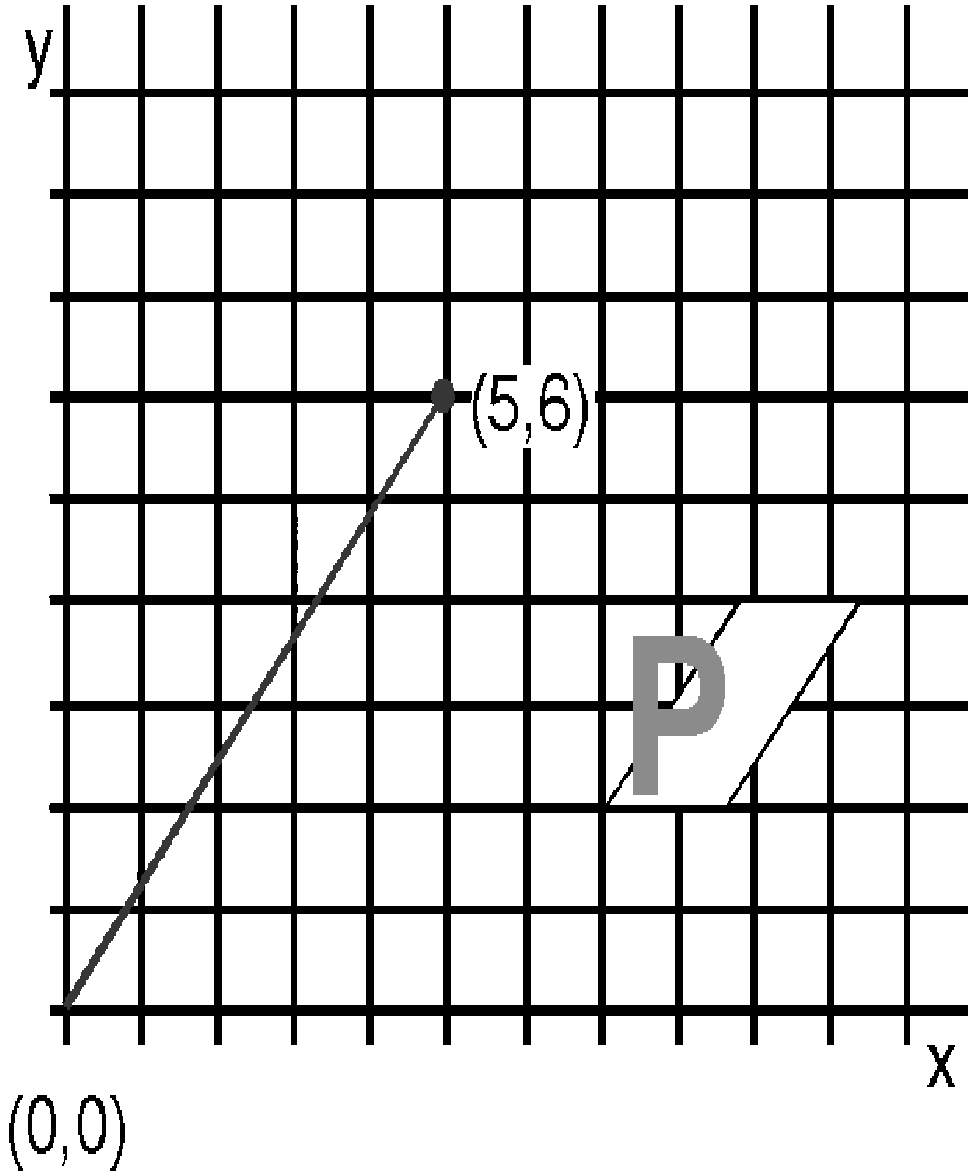
Each of the characters in the string in the previous figure is drawn with the vertical sides of its character cell parallel to the y-axis, and with the horizontal sides of its character box parallel to the x-axis.

Character Shear

Character shear is the angle defined by the x-axis and a vector drawn through the origin to a specified point in a Cartesian coordinate system. Neither (0,0) nor the specified point need have any relation to the current position. The operating system then aligns the vertical sides of the character cell. If the font is an outline font, the operating system also aligns the vertical strokes of the characters with the vector, regardless of the current character mode, as shown in the following figure.

An application can retrieve the point that defines the character shear vector using GpiQueryCharShear. An application can set the character shear using GpiSetCharShear, which accepts as input the x- and y-coordinates of a point that defines the new vector in a POINTL structure.

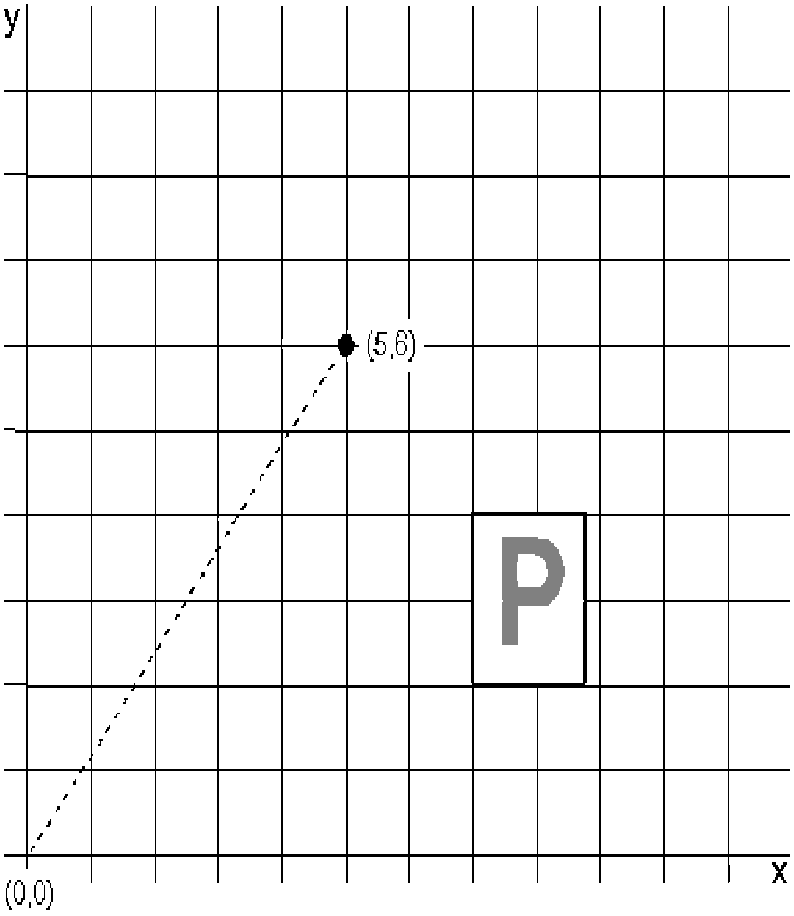
The *shear* of a character is the angle formed by the vertical lines of the character cell, and it can affect both the positioning and shape of characters in the character string. By default, the vertical lines of a character cell are parallel to the y-axis of the presentation page. As input to GpiSetCharShear supply the coordinates of the end point of a vector drawn from the origin (0,0). The effects of the current character shear value vary, depending on the current character mode and font type.



Effect of Character Shear on an Outline Font

The character cell is drawn with its vertical lines parallel to the vector from the origin to (5,6). The character is sheared to the same degree.

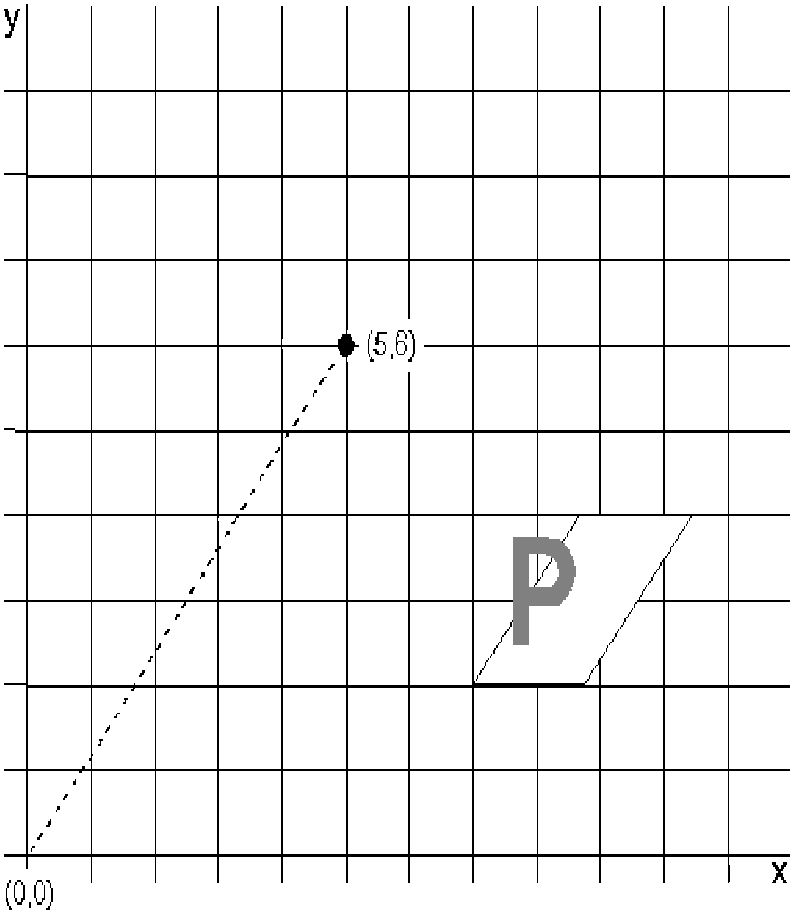
The character-shear value is ignored if the current font is an image font and the current character mode is CM_MODE1, as shown in the following figure.



Effect of Character Shear on an Image Font in CM_MODE1

The character string is unaffected by the character-shear value.

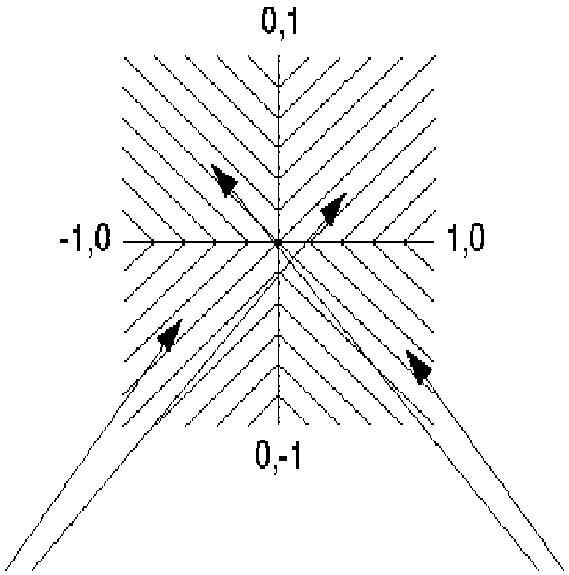
The character-shear value affects the positioning of the characters from an image font in CM_MODE2 only if character direction is CHDIRN_TOPBOTTOM or CHDIRN_BOTTOMTOP. That is, characters drawn vertically do not appear in a vertical line for nonzero shear angle from the vertical. The characters themselves cannot be sheared, as shown in the following figure.



Effect of Character Shear on an Image Font in CM_MODE2

The character cell is sheared, and it controls the positioning of the characters; the characters themselves are unchanged.

If the x- and y-coordinate values you specify in GpiSetCharShear are both positive and negative, the characters slant from lower left to upper right. If you supply one negative and one positive value, the characters slant from upper left to lower right, as illustrated in the following figure.



Effect of X- and Y-Values on Character Shear

Character shear, like other attributes, has a default value that can be changed using GpiSetDefAttrs (CBB_SHEAR). To reset the character shear to its default effect of drawing the vertical lines of the character cell parallel to the y-axis, supply a coordinate value of (0,1) on GpiSetCharShear.

Character Direction

Character direction is the direction in which the characters in a string are drawn in relation to the baseline. By default, the characters are drawn from left to right. An application can change the direction using GpiSetCharDirection. GpiSetCharDirection accepts as input one of the four values illustrated in the following figure. The value CHDIRN_DEFAULT is identical to CHDIRN_LEFTRIGHT.

```
  A   B   C                C   B   A
.           .            .           .
start       end          end         start

CHDIRN_LEFTRIGHT         CHDIRN_RIGHTLEFT


start                    end
  ¸                        ¸
  A                        C
  B                        B
  C                        A
```

∘	∘
end	start
CHDIRN_TOPBOTTOM	CHDIRN_BOTTOMTOP

An application can determine the current character direction using GpiQueryCharDirection.

Character Text Alignment

Character text alignment is the attribute that describes how the character strings are drawn with respect to the boundary of the output, either the current position or the starting position of the string, if a GpiCharString...At function draws the string. The alignment is set using GpiSetTextAlignment, which accepts as input a long value for horizontal and vertical alignment.

The acceptable values for GpiSetTextAlignment depend on the direction of the current coordinate system, as follows:

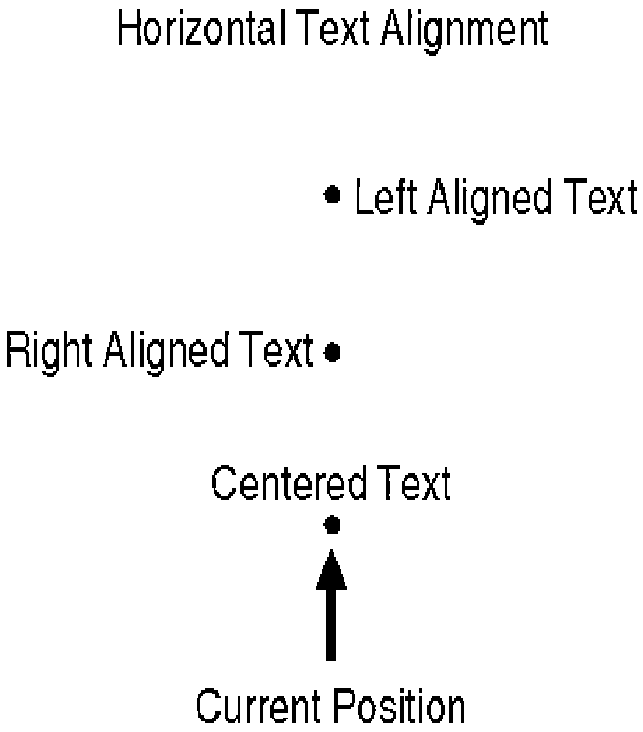
Value	Corresponds to the direction of...
Left	The lowest x-coordinate value
Right	The highest x-coordinate value
Top	The highest y-coordinate value
Bottom	The lowest y-coordinate value

Internally, the PM determines a reference point within a character string that is to be positioned over the starting point specified for the string.

If an application draws the string with either GpiCharString or GpiCharStringPos, the starting point specified for the string is the current position. If the application draws the string with either GpiCharStringAt or GpiCharStringPosAt, the starting point specified for the string is accepted as input by the function.

Horizontal Alignment of a Character String

When a horizontal character string does not fill the width of the output area, it can be positioned in one of the three ways shown in the following figure. All of these options can be set directly with the **lHorizontal** option of GpiSetTextAlignment.



Horizontal Positioning of Text Strings

Text justification requires an application to perform both queries and coordinate calculations. The following flags are used to specify types of horizontal alignment:

Horizontal Alignment Values

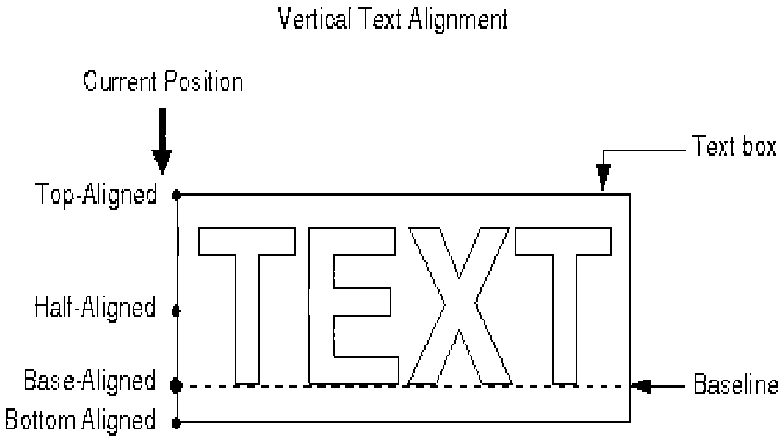
Identifier	Alignment
TA_LEFT	On the left edge of the leftmost character in the string
TA_RIGHT	On the right edge of the rightmost character in the string
TA_CENTER	On the arithmetic mean of the leftmost and rightmost characters in the string

There are two sets of default values for the **lHorizontal** option. They are provided for compatibility and map into the horizontal alignment values described above.

GpiQueryGraphicsField, GpiQueryPageViewport, and GpiQueryViewingLimits all provide methods of determining the width of the output area so your application can specify coordinates properly for the current position.

Vertical Alignment of a Character String

When a character string is to be displayed vertically, it can be positioned in one of the four ways illustrated in the following figure. The vertical options all can be set directly using the **lVertical** option of GpiSetTextAlignment.



Vertical Positioning of Text Strings

The following flags are used to specify types of vertical alignment:

Vertical Alignment Values

Identifier	Alignment
TA_TOP	On the top edge of the topmost character in the string
TA_HALF	On the arithmetic mean of the topmost and bottommost characters in the string
TA_BASE	On the baseline of the bottommost character in the string
TA_BOTTOM	On the bottom edge of the bottommost character in the string

There are two sets of default values for the **lVertical** option. They are provided for compatibility and map into the vertical alignment values described above.

Character Extra and Break Extra

Certain output devices permit you to specify extra space between character cells by using the *character extra* attribute. Sometimes the space between words can also be expanded by increasing the size of the *break character*, usually defined as the space character, by using the *break extra* attribute. If this adjustment to either attribute is permitted, the result will be in addition to the sizing effects caused by the following parameters:

- Font kerning
- Font proportional spacing
- Width vectors

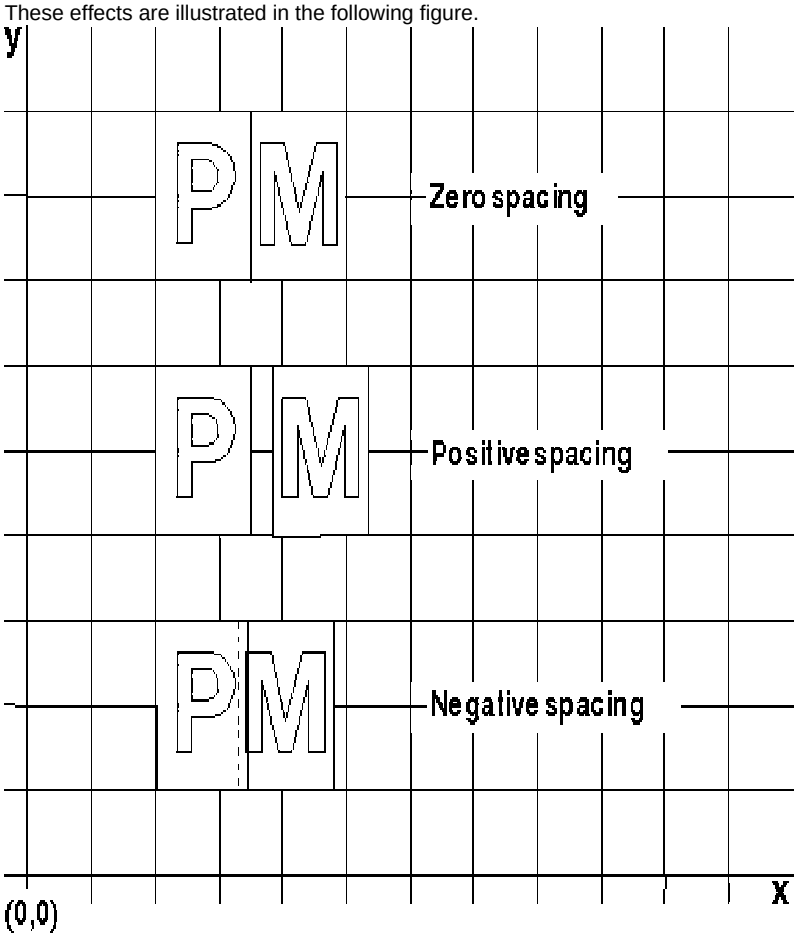
The break values you can specify create different effects, as follows:

Break Values and Their Effects

Value	Effect
Positive	Forces characters apart.

0	Resumes the default spacing created by other parameters.
Negative	Forces characters together, even overlapped characters.

These effects are illustrated in the following figure.



The Cumulative Effect of Break Values

The above are fixed integer values specified in world coordinates. Both the character extra and break extra attributes have initial default values of 0, which can be changed using GpiSetDefAttrs (CBB_EXTRA) and (CBB_BREAK_EXTRA). These values can be changed using GpiSetCharExtra or GpiSetCharBreakExtra respectively, and the values can be queried using GpiQueryCharExtra and GpiQueryCharBreakExtra.

Character Color and Mix

The *color* attribute defines the color used to draw a primitive or an object. The *mix* attribute determines how the color of a primitive or an object is combined with the color of the drawing surface or any other objects on the surface.

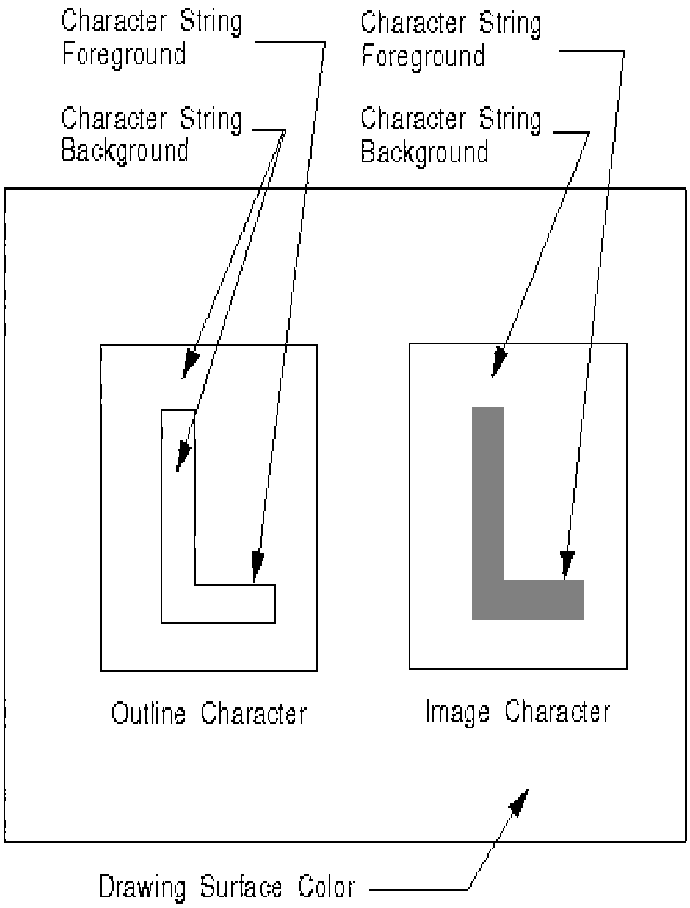
The character-string color defines the color used to draw the output from any of the draw-character-string functions. When a presentation space is created, the character-string color default is black. Character strings are one of the primitives that have both a foreground and a background color, as shown in the following figure.

For image characters, colors are determined by setting pels. For outline characters, the foreground consists of arcs and lines that define the character; the background color appears between the foreground lines. The character can be solid or filled, in which cases the background

color does not appear between the foreground lines.

Character string primitives have a color attribute for both the actual character and its character cell, which surrounds the character. The character-cell color is the background color.

The foreground mix attribute controls the combination of character-string color and drawing-surface color, while the background mix attribute controls the combination of the character-cell color and the drawing-surface color, as illustrated in the following figure.



Character String Primitives

When a presentation space is created, the character string mix attribute default is FM_OVERPAINT. The *overpaint* mix attribute specifies that the character-string color is not to be modified by the color of the drawing surface. If the character string mix attribute is changed, the character-string color is mixed with colors that are already on the drawing surface.

The character string background color default is CLR_BACKGROUND, usually defined by the application as the same color as the drawing surface. The character string background mix attribute default is BM_LEAVEALONE. The *leave-alone* background mix attribute specifies that the character string background color not be drawn. The cell that surrounds the character string appears only if the background character-string color and mix attributes are changed.

Use GpiSetAttrs to specify a new color or mix attribute. As input, this function accepts the following:

- Type of primitive, for example, PRIM_CHAR
- List of attributes to be changed
- List of attributes to be set to their default values
- Values for the attributes to be changed

GpiSetAttrs also is useful to specify color and mix attributes for a specific data structure -for example, CHARBUNDLE. GpiSetAttrs provides some protection against invalid colors.

To determine the current character-string color and mix attributes, call GpiQueryAttrs, which accepts as input the primitive type and the attributes in question. GpiQueryAttrs returns an array of values for the queried attributes.

To reset the default character-string color and mix attributes, as with all attributes specified in CHARBUNDLE, call GpiSetDefAttrs, which accepts as input the type of primitive, attributes to be changed, and values that will become the new default values. Changing default values

is especially important when working with segments. Changing the default values during a series of drawing functions is not recommended.

The character color and mix attributes also can be specified using the following functions:

- GpiSetColor
- GpiSetMix
- GpiSetBackColor
- GpiSetBackMix

If the character color, character background color, mix, or background mix attributes are specified individually, the following queries can return a value inconsistent with the current character attributes:

- GpiQueryColor
- GpiQueryMix
- GpiQueryBackColor
- GpiQueryBackMix

Using Character String Primitives

You can use the character string primitive functions to perform the following tasks:

- Draw a string of characters using the selected font.
- Modify the string by using one or more of the following operations:

 Scaling
 Shearing
 Rotating
 Transforming
 Changing the angle of the baseline

- Aligning text

Drawing Text

The following figure shows how to select a Helvetica outline font, set the size of the character box, change the foreground color to red, set the character angle, and move the cursor to a specified location. Then, GpiCharString is used to write a string of characters with the specified size, color, angle, and location.

```
#define INCL_GPIPRIMITIVES
#define INCL_GPILCIDS
#include <os2.h>
void fncFONT09(void){

    POINTL ptl = { 100, 50 };
    GRADIENTL grad = { 4, 1 };
    SIZEF sizfx;
    FATTRS fat;
    CHARBUNDLE cbnd;
    FONTMETRICS afm[80];
    HPS hps;
    HDC hdc;
    LONG cHelvFonts, i;
    LONG cFonts = 0;
    LONG lcid = 1;
    LONG devRes[2];              /* Horizontal, vertical font resolutions */

    cHelvFonts = GpiQueryFonts(hps, QF_PUBLIC, "Helv",
        &cFonts, sizeof(FONTMETRICS), NULL);
```

```
    GpiQueryFonts(hps, QF_PUBLIC, "Helv", &cHelvFonts,
        sizeof(FONTMETRICS), afm);
                                /* Find an outline Helvetica font.       */

    for (i = 0; (!(afm[i].fsDefn & FM_DEFN_OUTLINE)) &&
               i < cHelvFonts; i++)    ;

    fat.usRecordLength = sizeof(FATTRS);
    fat.fsSelection = 0;
    fat.lMatch = afm[i].lMatch;
    StringCopy(fat.szFacename, afm[i].szFacename);
    fat.idRegistry = 0;
    fat.usCodePage = 0;
    fat.lMaxBaselineExt = 0;
    fat.lAveCharWidth = 0;
    fat.fsType = 0;
    fat.fsFontUse = FATTR_FONTUSE_OUTLINE;

    GpiCreateLogFont(hps, (PSTR8) NULL, lcid, &fat);
    GpiSetCharSet(hps, lcid);

    DevQueryCaps(hdc, CAPS_HORIZONTAL_FONT_RES, 2L, devRes);
    sizfx.cx = MAKEFIXED((afm[i].sNominalPointSize * devRes[0])/ 720, 0);
    sizfx.cy = MAKEFIXED((afm[i].sNominalPointSize * devRes[1])/ 720, 0);
    GpiSetCharBox(hps, &sizfx);

    cbnd.lColor = CLR_RED;
    GpiSetAttrs(hps, PRIM_CHAR, CBB_COLOR, NULLHANDLE, &cbnd);
    GpiSetCharAngle(hps, &grad);
    GpiMove(hps, &ptl);
    GpiCharString(hps, 11, "Vector Text");
} /* fncFONT09 */
```

Certain parameters in the above example are explained in Fonts.

Formatting Text

Graphics text, like all other graphics objects, has to be positioned correctly in the output area, which usually consists of one of the following:

- Entire client area of a PM window
- Part of a PM window
- Addressable area of a printer

Unlike other graphics objects, however, text is governed by well-established readability and usability rules. These rules apply generally to text output, whatever its method of production. Following are some recommendations:

- Lines of text from fonts with large point sizes must be more widely spaced for maximum readability.
- The longer the line of text, the greater the space between lines must be to ensure that the lines do not appear to merge.
- Very small text must be split into multiple columns rather than continued across the page.

Some of these considerations are taken care of by the font designer. The PM enables you to control both the horizontal and vertical positioning of text.

Positioning Text in World Coordinates

When considering text alignment, take the versatility of the coordinate systems into account. The following definitions depend on the current coordinate system:

World Coordinate Values

Value	Corresponds to the direction of...
Left	The lowest x-coordinate value
Right	The highest x-coordinate value
Top	The highest y-coordinate value
Bottom	The lowest y-coordinate value

To position a character string horizontally, you must know the width of the output area and the length of the character string. The PM provides three different functions for determining the width of the output:

Function Name	Description
GpiQueryGraphicsField	Returns the bottom-left and top-right corners of the graphics field in presentation page units.
GpiQueryViewingLimits	Returns the viewing limit.
GpiQueryPageViewport	Returns the page viewport in device units.

GpiConvert changes coordinates into world coordinates. To calculate the width of the output area, subtract its **left** from its **right**. For example, if the **left** is 30, and **right** is 600, the width of the output area is 570 world coordinates.

The PM provides three different functions for determining the length of the character string primitive:

- GpiQueryTextBox
- GpiQueryCharStringPos
- GpiQueryCharStringPosAt

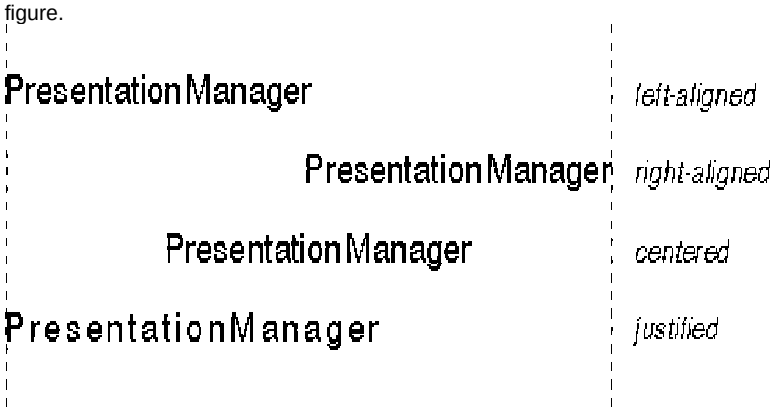
GpiQueryTextBox returns the relative coordinates of a parallelogram that surrounds the character string. By subtracting the x-coordinate of TXTBOX_BOTTOMRIGHT from the x-coordinate of TXTBOX_BOTTOMLEFT, an application can determine the length of the string.

GpiQueryCharStringPos returns an array of points, in which the world coordinate position of each character in the string is recorded. The last value in the array becomes the new current position if the string is drawn using GpiCharStringPos. By subtracting this position from the current position (obtained using GpiQueryCurrentPosition), the length of the string can be determined.

GpiQueryCharStringPosAt also returns an array of points, in which the world coordinate position of each character in the string is recorded. The last value of the array becomes the new current position if the string is drawn using GpiCharStringPosAt. This function accepts a specified starting position for the character string. By specifying a starting position of (0,0) for example, an application can determine the length of the string without subtraction.

The current position actually is not updated by either GpiQueryCharStringPos or GpiQueryCharStringPosAt.

When a character string does not fill the width of the output area, it can be positioned in one of the four ways illustrated in the following figure.



Horizontal Positioning of Text Strings

To left-align the text, set the x-coordinate of the current position to the *left* of the output area before drawing the character string. GpiSetCurrentPosition must be used to set the current position if your application draws the string using either GpiCharString or GpiCharStringPos. Both GpiCharStringAt and GpiCharStringPosAt accept a starting position as input.

To right-align the text, subtract the length of the character string from the width of the output area, then add the difference to the x-coordinate of the current position before drawing the character string. If the output area is 570 world coordinates wide, for example, and the character string is 436 world coordinates long, add 134 to the x-coordinate of the current position before drawing the text.

To center the text, subtract the length of the character string from the width of the output area, then divide the difference by 2 before adding it to the x-coordinate of the current position. If the difference is 134, for example, you add 67 to the x-coordinate of the current position.

To justify the text, so that the text string fills the width of the output area, distribute the surplus space throughout the character string. For example, you could add the extra space to the break characters only, or you could share it equally among all characters in the string. Text justification requires the individual positioning each character in the string using either of the following calls:

- GpiCharStringPos-draws at the current position and permits you to position every character after the first.
- GpiCharStringPosAt-permits you to position every character, including the first.

Both functions enable you to specify a different character increment for each character in the string. Distance is measured from the character reference point of one character to the character reference point of the next character. The values you specify apply only to the character string supplied at input; they *do not* become current attribute values.

If you are formatting a block of text, the string might be wider than the output area, or longer than 256 characters. In either case, your application must separate the string into smaller groups of characters so that it fits either criteria. A good starting point is to determine the number of characters planned for each line. Dividing the output width by the character cell width can provide a first estimate as to where to separate a character string. An application can use this estimate to work through the string looking for spaces. Each time you find a space, compare the length of the string (up to the space) with the width of the output area. When the string is longer than the output area, work back to the previous space and display or print that part of the string. You can format the entire block of text for the output area in this way.

When you are formatting a block instead of a single line of text, an application must specify the vertical placement of each line. If you are using an image font, you have the assurance that each character is the same height. However, you do not have that assurance with an outline font, nor that the text block is written in the same font. Therefore, when calculating the separation of lines, avoid using complex combinations of font metrics values. Instead, it is recommended that you multiply the desired point size, or em-height metric, of the text by 1.2.

Clipping and Boundary Determination

Both clipping and boundary determination are graphics operations concerned with limiting the amount of graphics information passed between different coordinate spaces. The following topics are related to the information in this chapter:

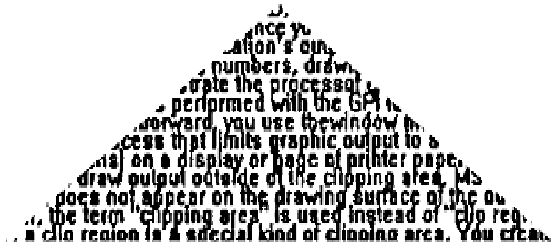
- Presentation spaces and device contexts
- Coordinate spaces and transformations
- Regions
- Paths

About Clipping

Clipping enables the PM to discard parts of a picture that lie outside a specified *clipping boundary* . The parts of the picture enclosed by the boundary are said to be inside the *clipping area* . A clipping area might be a single rectangle or a complex shape, depending on the method used to define it.

Note: In this chapter, the word *area* does not refer to an *area primitive* ; it describes the shape or shapes used for clipping graphics output.

If an application attempts to draw outside of a clipping area, the operating system ensures that the output does not appear on the drawing surface of the output device. For example, if an application defined a triangular clipping area before drawing text output, all text outside of the triangle would be discarded, even though an entire page of text was defined. The following figure illustrates the result.



Triangular Clip Path

Clipping boundaries are defined by the application. The PM performs some clipping automatically when, for example, your application's graphic output is clipped to fit a client window area or the device's output area for a hardcopy device.

Types of Clipping Areas

The PM programming interface assembles the application's graphics output in a process that can use up to five coordinate spaces, known collectively as the *viewing pipeline* . Clipping can occur at each stage in the viewing pipeline. Objects in world coordinate, model, page, or device space can be clipped. When an application defines clipping areas in several coordinate spaces, the final result is similar to combining all the areas into a single clipping area. This single area is defined by the intersection of the areas in each coordinate space.

Clipping in different coordinate spaces, however, is a means of *conceptualizing* the process to aid in its understanding. Clipping, like transformations, actually happens in one operation, with all the different types of clipping being performed on all primitives in the device space at once.

The following table describes the different types of clipping areas associated with the different coordinate spaces.

Clipping Areas and Coordinate Space Summary

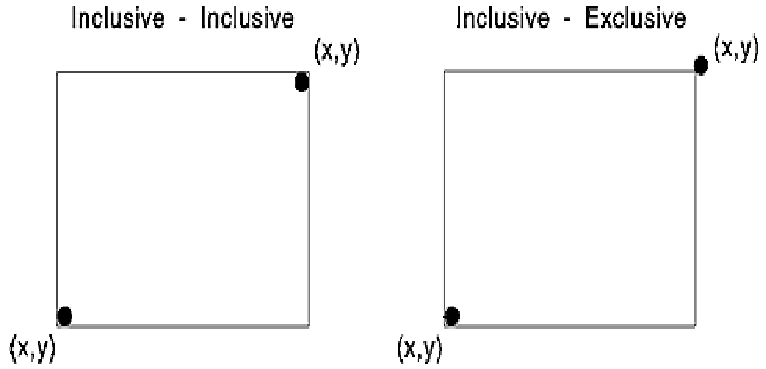
Clipping Area	Coordinate Space	Description
Clip path	World space	Always inclusive/inclusive Clipping area can have curved edges Clipping area can be rotated.
Viewing limit	Model space	Always a rectangular clipping boundary Always inclusive/inclusive Rotating clipping area results in larger rectangle.
Graphics field	Page space	Always a rectangular clipping boundary Always inclusive/inclusive Rotating clipping area impossible. Cannot specify a device transform with rotation.
Clip region	Device space	Can be a single rectangle or multiple rectangles that overlap or remain separate

```
Always inclusive/exclusive

Rotating the clipping area
is impossible.
```

Note: *Inclusive/inclusive* means that the operating system includes the bottom and leftmost edges of the rectangle in the clipping area as well as the top and rightmost edges. *Inclusive/exclusive* means that the operating system includes the bottom and leftmost edges but excludes the top and rightmost edges.

The following figure illustrates the differences between inclusive-inclusive and inclusive-exclusive clipping.



Inclusive/Inclusive and Inclusive/Exclusive Clipping

The Clip Path

The *clip path* defines the clipping boundary in world coordinate space. Paths are graphic objects that serve many purposes, only one of which is to be used as a clipping mechanism. Paths are described in Paths. The recommended functional sequence for creating a clip path is as follows:

Function	Effect
GpiBeginPath	Begins the path definition.
Line and arc GPIs	Gives the path shape.
GpiEndPath	Ends the path definition.
GpiModifyPath	An optional step explained below.
GpiSetClipPath	Converts the path to a clip path.

Before converting to a clip path, the path can be modified using GpiModifyPath. If modified, the path is converted to a geometric (wide) line using the current geometric width, line join, and line end attributes. The shape defined by the geometric line then is used for the clip path. The clip path can be a simple unmodified path.

GpiSetClipPath accepts two different path identifiers as input:

- 1SCP_RESET (default)
- 0SCP_AND

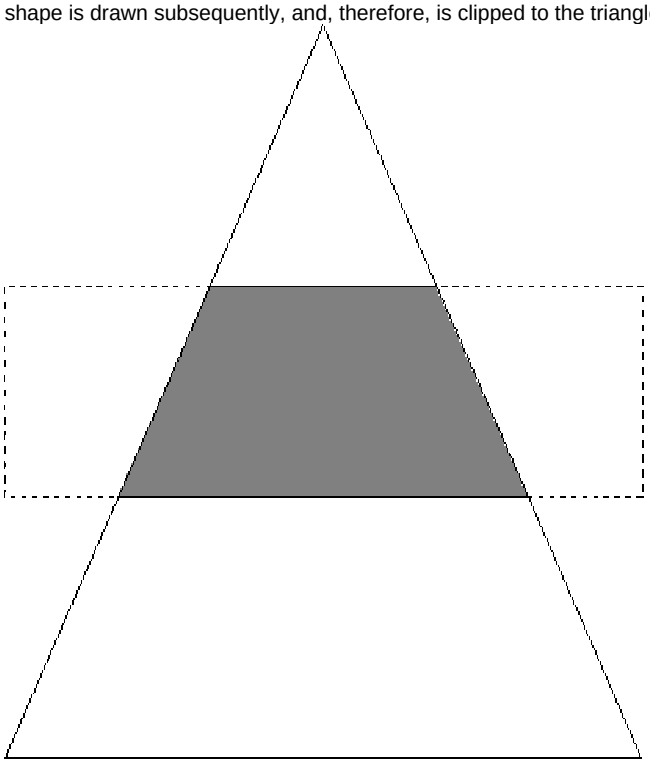
The default path identifier, SCP_RESET, resets the clip path to infinity, which displays the picture without clipping. If this value is selected, the current clip path definition is discarded instead of stored.

A path identifier of SCP_AND specifies that the clip path be redefined as the mathematical intersection of the stored clip path and the

current path definition.

The only method of specifying the clip path to the current path, after GpiSetClipPath has been called, is to call GpiSetClipPath twice: the first call with a path identifier of SCP_RESET, and the second with a path identifier of SCP_AND.

The following figure shows a triangle shape that has been defined within a path bracket and selected as the current clip path. The filled box shape is drawn subsequently, and, therefore, is clipped to the triangle.



The Clipping Path

The broken lines show the area of the box that has been clipped.

Clip paths are most useful when you want to use an irregular clipping boundary, or when the clipping boundary itself is an integral part of the picture. Both are true of the clip path in the previous figure.

GpiSetClipPath also accepts one of two construction options as input:

- SCP_ALTERNATE (default)
- SCP_WINDING (must be selected if path has been modified).

Any drawing that is clipped to the current clip path must follow the alternate or the winding rules as to whether that portion of a picture is included in the clip area. The alternate and winding modes are described for paths in Paths in Alternate Mode and Paths in Winding Mode. Any point on the boundary of the path is considered within the path and is not clipped.

To end clipping to the current clip path, call GpiSetClipPath with an identifier of 0. This function deselects the current clip path by setting it to infinity. In some circumstances, the current clip path is deselected automatically.

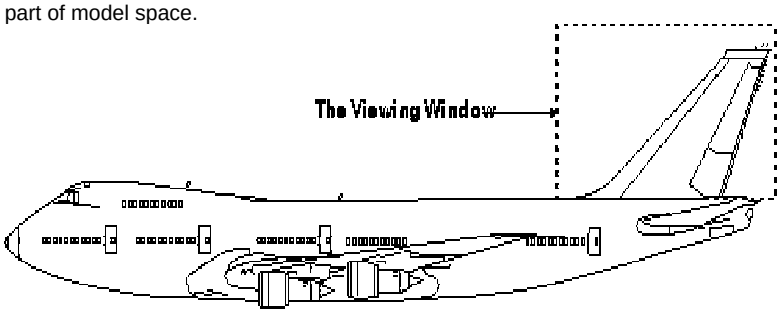
A path definition can be stored in a graphics segment, and if that segment is retained, the path can be re-created as required when the segment is redrawn. Clip path definitions can be stored in a retained segment also and redrawn when required. In *draw-and-retain* mode (DM_DRAWANDRETAIN), the initial path or clip path is created as the segment is constructed. If the current drawing mode is *retain* (DM_RETAIN), however, the path or clip path is not created until the first time the segment is drawn.

The Viewing Window

The *viewing window* defines a rectangular clipping boundary in model space. It is defined with GpiSetViewingLimits. As input to this

function, you supply the model coordinates of the lower-left and upper-right corners of the viewing window.

When a drawing primitive, such as a line, intersects a viewing window, any part of that line outside of the viewing window is clipped. Any point on the boundary of the viewing window is considered within the window, and is not clipped. By default, the viewing window performs no clipping. In this case, all graphics output in the model space is transformed. The following figure shows how the viewing window outlines a part of model space.



The Viewing Window

This example shows how a presentation page is constructed. The viewing window outlines the tail of the aircraft, which is scaled and translated when drawn in presentation-page space. The rest of the aircraft is clipped away during the drawing process.

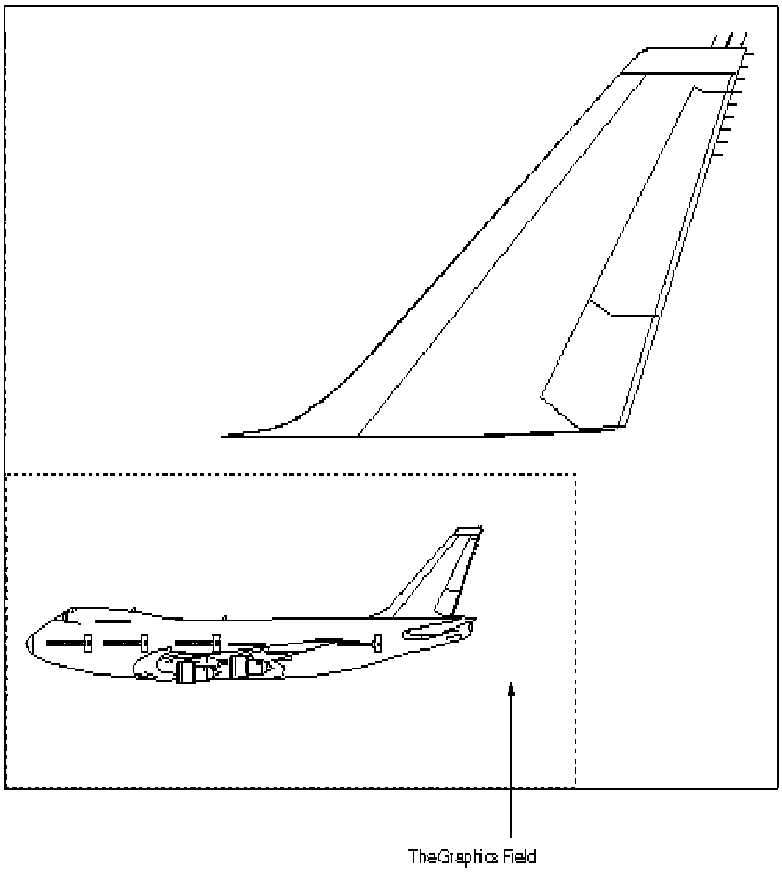
The Graphics Field

The graphics field defines a rectangular clipping area in the presentation page. It is defined in GpiSetGraphicsField.

Note: If this clipping area is to be used, it must be defined before any drawing begins.

Specify the size of the graphics field in presentation page coordinates as input to GpiSetGraphicsField.

Only the graphic output contained in this clipping boundary is visible when the presentation page is transformed to device space. By default, the graphics field is infinitely large and therefore performs no clipping. If you do specify a graphics field, however, any point on its boundary is considered within the graphics field and is not clipped. When a drawing primitive, such as a line, intersects a graphic field, any part of the line outside the graphics field is clipped. The following figure shows how a graphics field could be defined for a presentation page.



The Graphics Field

The broken line shows an arbitrary graphics field that is smaller than the presentation page. The aircraft tail, a separate object in the presentation page, is outside the graphics field and is clipped away as it is drawn.

The picture assembled in the graphics field is the picture that is displayed or printed. If you do not define a graphics field, the picture assembled in the presentation page is the picture that is displayed or printed. The presentation page is not a clipping boundary, and graphics in page coordinate space that are outside the presentation page boundary, therefore, might be visible.

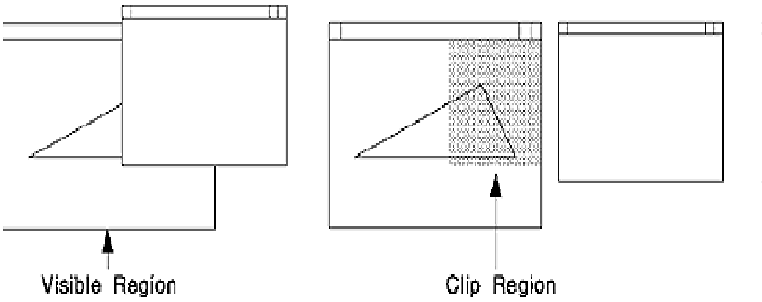
The Current Clipping Region

Clipping regions are clipping areas defined (as regions) by one or more rectangles in device coordinates. Because they are defined in device coordinates, clipping regions do not suffer from the rounding errors associated with other types of clipping. Therefore, they are ideally suited to redraw part of the picture without boundary discontinuities, for example, after a BitBlt operation has been used to scroll a picture in a window.

Regions are not available automatically for clipping. To select an existing region as the current clipping region, use GpiSetClipRegion. By default, the clipping region is the same size as the drawing surface. Only one clipping region can exist in the presentation space at one time. To end clipping to the current clipping region, deselect it by calling GpiSetClipRegion with a NULL region handle. A deselected clipping region retains the effects of any changes made to it while it was a clipping region and can be reselected.

You do not have to deselect the current clipping region before selecting another. Each selected clipping region automatically replaces the one before it. If a clipping region exists when you call GpiSetClipRegion, the existing clipping region reverts to being a normal region, and its handle is returned.

Clip paths and clip regions share a common implementation, but clip regions are faster to create than clip paths. This might be a performance factor when designing your application for repairing the screen or redrawing the picture in a client window after the display has changed. The following figure illustrates this use of regions.



Screen Repairing

Note: GpiSetClipRegion does not cause graphics orders to be added to the current segment. Therefore, variations in the clip region must not be used to construct the picture. The clip region is intended to define a fixed clipping area for the entire picture.

When you select the current clipping region, none of the region-related GPI functions can be used for that region. The PM provides a series of functions that *mirror* the region-related functions. However, all of these functions work in world coordinates rather than device coordinates, and, therefore, are subject to current transformations.

Any of the following functions can be used to get information about or to redefine the current clipping region.

- **GpiQueryClipBox**

 You can request the dimensions of the smallest rectangle that encloses all current clipping boundaries by calling GpiQueryClipBox. The following boundaries are included in this calculation:

 - Current clip path
 - Current viewing window
 - Current graphics field
 - Current clipping region
 - Visible region of the window

- **GpiIntersectClipRectangle**

 GpiIntersectClipRectangle redefines the current clipping region to the intersection of the existing clipping region with the rectangle whose dimensions you supply in this function. This has the same effect as CRGN_AND in GpiCombineRegion.

- **GpiExcludeClipRectangle**

 You also can redefine the current clipping region using GpiExcludeClipRectangle. This function excludes a specified rectangle from the current region and has the same effect as CRGN_DIFF on GpiCombineRegion.

- **GpiOffsetClipRegion**

 The current clipping region can be moved from its current position using GpiOffsetClipRegion.

- **GpiPtVisible**

 GpiPtVisible tells you whether a point, expressed in world coordinates, is visible on the screen. A point is visible if it is within all current clipping boundaries and is in the visible region of the window.

- **GpiRectVisible**

 GpiRectVisible tells you whether any part or the whole of a rectangle, whose dimensions you supply in world coordinates, is visible on the screen. The rectangle is visible if it intersects both the visible region of the window and all current clipping boundaries.

How Clipping Is Implemented

The rules by which the PM implements clipping are as follows:

- Any primitive completely outside the clipping boundary is discarded.

 When a primitive crosses a clipping boundary, any part outside the boundary is discarded.

- Any primitive completely within the clipping boundary is retained.

 When a primitive crosses a clipping boundary, any part within the boundary is retained.

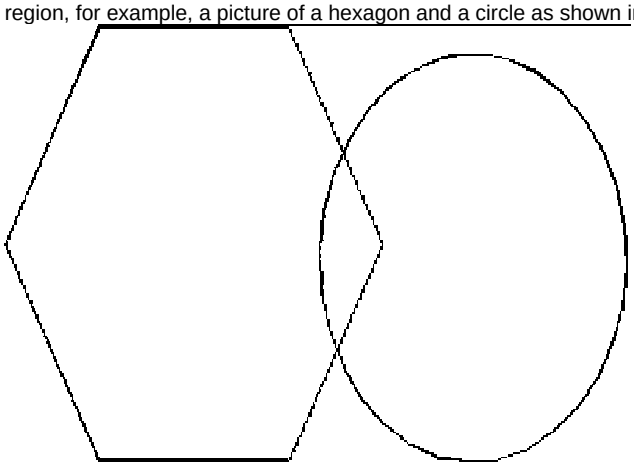
- When the clipping boundary is a clip path, viewing window, or graphics field, any point that falls on the boundary is considered within the clipping boundary.

 When the clipping boundary is a clipping region, any point on the top or right boundaries of a rectangle is discarded; and any point on the bottom or left boundaries of a rectangle, that is not on the right or top boundaries also, is included in the region.

Redrawing Nondynamic Graphics

An interactive graphics application usually permits changes to the displayed picture. For example, an object can be moved or sized, and you can plan for this by defining particular segments as *dynamic* . Dynamic segments are described in Editing Retained Graphics and Graphics Segments.

If dynamic segments are inappropriate (when you are using nonretained graphics, for example), you can repair the picture using a clipping region, for example, a picture of a hexagon and a circle as shown in the following figure.

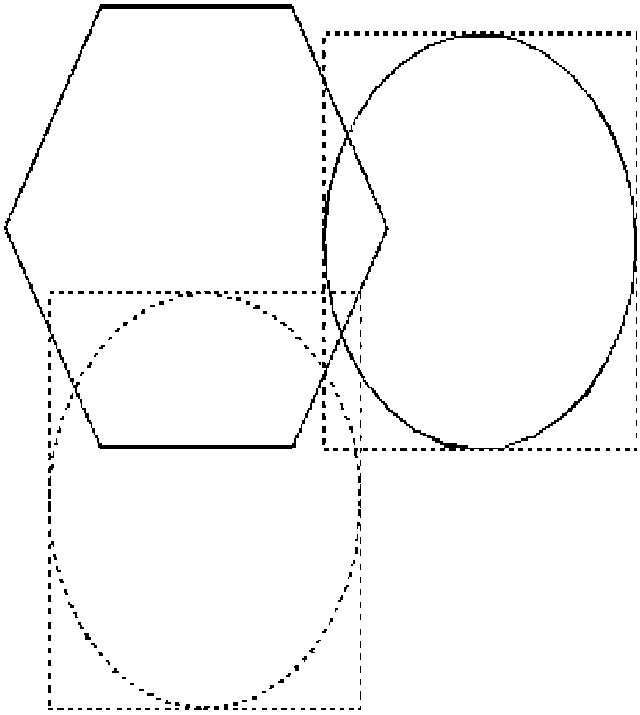


A Hexagon and Circle

If the circle is moved to another screen position by the use of an input device, you must repair its original location and redraw it in its new location. The following figure shows this sequence of events. Following are the steps required to do this:

1. Determine the size of the smallest rectangle that contains the circle in its current position using a process called *boundary determination* .
2. Switch off the DCTL_DISPLAY flag of GpiSetDrawControl, apply a translation transformation to the circle, and redraw it in its new position.
3. Determine the size of the smallest rectangle that contains the circle in its new position using boundary determination.
4. Use GpiConvert to convert the model-space coordinates provided by the boundary-determination process to device-space coordinates.
5. Use the device-space coordinates of the two rectangles to create a region, and select it as the current clipping region.
6. Switch on the DCTL_DISPLAY flag of GpiSetDrawControl.
7. Call GpiErase (or set the erase-before-draw control) to erase the current contents of the clipping region.
8. Redraw the picture with the circle in its new position. Any part of the picture within the clipping region is redrawn. That part of the

hexagon that is outside the clipping region is unaffected by the change and does not have to be redrawn.



Defining a Clipping Region

The broken circle shows the position to which the circle is to be moved. The two bounding rectangles overlap, and produce a complex region. If the circle were to be moved much farther away from its start position, the region would comprise two disjoint rectangles.

About Boundary Determination

Boundary determination is an operation to compute the size of the smallest rectangle that encloses a graphics output in model space. One use of boundary determination is to enable you to repair only the affected parts of the screen, when a graphics object is moved, for example, or when a graphics object is changed some other way. Dynamic segments are not included in boundary-determination operations.

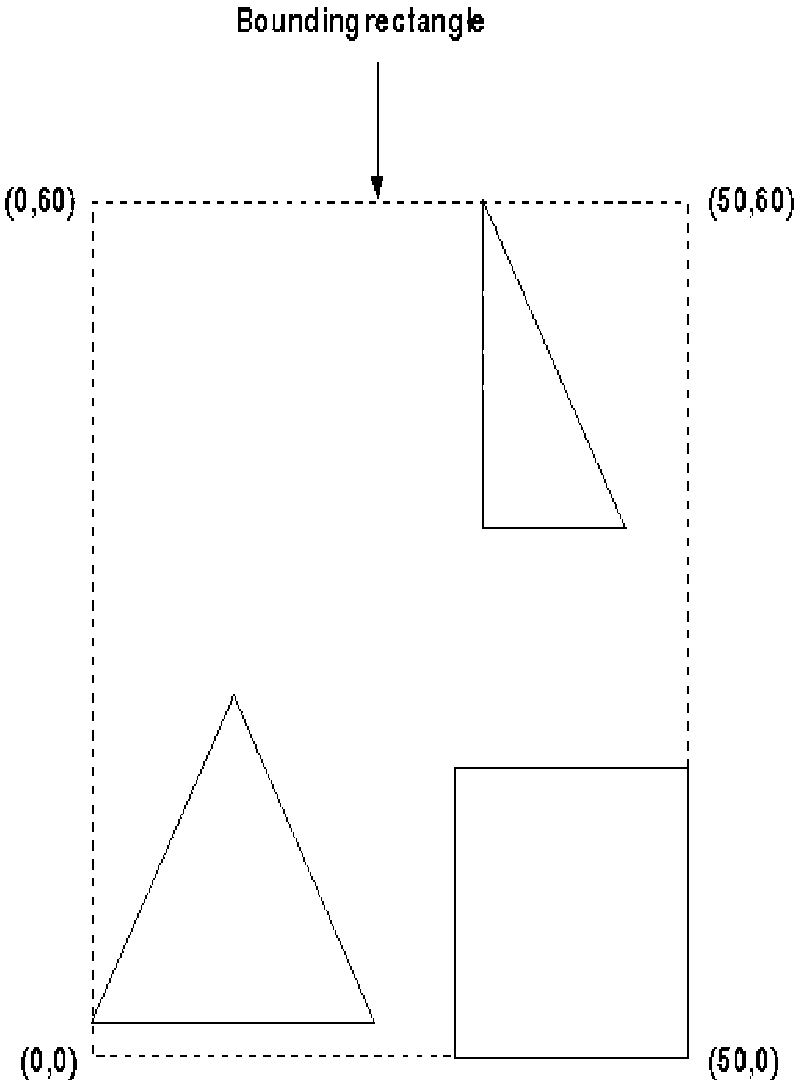
Boundary determination can be performed on both retained and nonretained graphics. In both instances, you request boundary data to be calculated by setting the *boundary data flag* (DCTL_BOUNDARY) in GpiSetDrawControl. If you do not set this flag (for example, if you do not want to collect boundary data unnecessarily) and later find that you need boundary data for a particular object, you can do the following:

1. Switch on the boundary-data flag, and switch off the display flag, using GpiSetDrawControl.
2. Redraw the object in its current location. Boundary data is collected, but the object is unaltered.

If you are drawing retained graphics, each drawing request (GpiDrawSegment, GpiDrawFrom, and GpiDrawChain) causes the boundary data resulting from the drawing to be made available. The application must request this data explicitly by calling GpiQueryDeviceBitmapFormats after each drawing request for which it wants to examine boundary data. Boundary data is returned to the application in model space coordinates. The boundary data is reset before each retained drawing operation, so there is no risk of accumulating data from separate operations.

If you are drawing nonretained graphics, boundary data is accumulated for each GpiPutData and for each individual primitive drawing function. The application can request the accumulated boundary data at any time by calling GpiQueryBoundaryData. Data continues to accumulate unless you call the GpiResetBoundaryData; it is not reset automatically.

The boundary data returned to you is in the form of four model-space coordinates, which are the lowest (x,y) positions and the highest (x,y) positions of the bounding rectangle in model space as illustrated in the following figure.



The Bounding Rectangle

Using Clipping and Boundary Determination

This section explains how to use clipping functions to:

- Create a clip path or clip region
- Exclude a rectangular area from a clip region
- Add a rectangular area to a clip region
- Set the clip region to the intersection of the current clip region and a specified rectangle
- Determine the size of the smallest rectangle that will surround the intersection of the current clipping areas completely

Creating a Clip Path

A drawing and computer-aided-design (CAD) application may require the ability to clip to curved edges. If so, it must use a clip path to define a curved clipping area in world coordinates. Because clip paths (especially ones that clip to curved edges) require considerable memory and processing time, use them only when necessary. Whenever possible, your application must use a clip region, graphics field, or viewing limit.

To create a clip path, do the following:

1. Determine the clip path's shape and size (in world coordinates).
2. Use GpiBeginPath to begin defining the path.
3. Create the path.
4. Close the path definition with GpiEndPath.
5. Create a clip path from the path definition with GpiSetClipPath.

The following figure uses this procedure to create an elliptical clip path.

```
#define INCL_GPIPATHS
#include <os2.h>
void fncCLIP01(void){
    HPS    hps;                     /* Presentation-space handle */
    POINTL ptl1;                    /* Point structure           */
    FIXED  fxArc;                   /* Multiplier for arc        */
    LONG idPath;                    /* Path identifier           */
        .
        .           /* Load ptl1 with coordinates of clip path.   */
        .
    idPath = 1;
    GpiBeginPath(hps, idPath);      /* Begins path               */
    GpiMove(hps, &ptl1);            /* Sets current position     */
    fxArc  = MAKEFIXED(50, 0);      /* Sets arc multiplier       */
    GpiFullArc(hps, DRO_OUTLINE, fxArc);  /* Defines ellipse     */
    GpiEndPath(hps);                /* Ends path                 */
    GpiSetClipPath(hps, idPath, SCP_ALTERNATE | SCP_AND);
} /* fncCLIP01 */
```

Creating a Clip Region

To create a clip region, first determine its size and shape (in device coordinates). Load the coordinates for the rectangles that define the clip region into an array of RECTL structures. Then create the region and set it to a clip region with GpiCreateRegion and GpiSetClipRegion respectively.

The following figure shows how to create a clip region.

```
#include <os2.h>
void fncCLIP02(void){
    HPS  hps;
    HRGN hrgn;
    RECTL arcl[3];
        .
        .  /* Load array of RECTL structures with coordinates.       */
        .
    hrgn = GpiCreateRegion(hps, sizeof(arcl) / sizeof(RECTL), arcl);
    GpiSetClipRegion(hps, hrgn, NULL);
} /* fncCLIP02 */
```

Excluding a Rectangular Area from a Clip Region

Some applications let you prepare output in multiple stages. For example, a word processing application might permit you to prepare your text first and then add bit maps that enhance and support the text. These applications can use GpiExcludeClipRectangle to exclude an area from a clip region, preventing the user from deleting output that already exists.

To exclude an area from a clip region, first determine the dimensions (in device coordinates) of the smallest rectangle that completely surrounds the area to exclude from the clip region. Then call GpiExcludeClipRectangle, including as input the dimensions of the area to exclude. The following figure illustrates these steps.

```
#include <os2.h>
void fncCLIP03(void){
    HPS   hps;
    RECTL rcl;
        .
        .  /* Set rectangle coordinates here. */
        .
    GpiExcludeClipRectangle(hps, &rcl);
} /* fncCLIP03 */
```

Adding a Rectangular Area to a Clip Region

Some applications might need to increase the size of a clip region. For example, a user might request that a desktop publishing application extend a column of text on a page.

To add a rectangular area to a clip region, follow these steps:

1. Determine the dimensions (in device coordinates) of the rectangular area to add to the current clip region.
2. Release the current clip region using GpiSetClipRegion. GpiCombineRegion cannot combine regions if either of the regions is a clip region.
3. Call GpiCreateRegion and pass it the dimensions of the rectangle that you defined in Step 1. This creates a second region that you can combine with the first.
4. Call GpiCreateRegion again and create a third region that will be the final destination region.
5. Call GpiCombineRegion to create a region that combines the original region and the region you created in Step 2. It is not essential to create a third region because GpiCombineRegion can use one of the 2 source regions being combined for a target region.
6. Call GpiSetClipRegion and pass it the handle returned by GpiCombineRegion.

The following figure illustrates these steps.

```
#define INCL_GPIREGIONS
#include <os2.h>
void fncCLIP04(void){
    HPS   hps;
    RECTL rcl1, rcl2, rcl3;
    HRGN  hrgn1, hrgn2, hrgn3;

    hrgn1 = GpiCreateRegion(hps, sizeof(rcl1) / sizeof(RECTL), &rcl1);
    GpiSetClipRegion(hps, hrgn1, NULL);       /* Creates first clipping region */
        .
        .  /* Compute coordinates of second region here.                    */
        .
    GpiSetClipRegion(hps, NULLHANDLE, NULL);  /* Releases first clipping region*/
    hrgn2 = GpiCreateRegion(hps, sizeof(rcl2) / Sizeof(RECTL), &rcl2);
    hrgn3 = GpiCreateRegion(hps, sizeof(rcl3) / Sizeof(RECTL), &rcl3);
    GpiCombineRegion(hps, hrgn3, hrgn1, hrgn2, CRGN_OR);
    GpiSetClipRegion(hps, hrgn3, NULL);       /* Creates second clipping region*/
} /* fncCLIP04 */
```

Setting the Clip Region to a Region Intersection

Your application might require the ability to set the clip region to the intersection of the current clip region and another region. Do this with GpiIntersectClipRectangle, as shown in the following:

```
#include <os2.h>
void fncCLIP05(void){
    RECTL rcl;
    HPS hps;
        .
        . /* Load rcl with coordinates of rectangle to intersect. */
        .
    GpiIntersectClipRectangle(hps, &rcl);
} /* fncCLIP05 */
```

Determining the Size of a Clipping Area

If an application is able to specify a clip path in world space, a viewing limit in model space, a graphics field in page space, and a clip region in device space, it might be necessary for your application to determine the size of the clipping area formed by the intersection of the four. GpiQueryClipBox returns the dimensions (in world coordinates) of the smallest rectangle that completely surrounds the intersection of all the defined clipping areas, including the visible region.

The following figure shows how to use GpiQueryClipBox to fill a RECTL structure with the desired coordinates.

```
#include <os2.h>
void fncCLIP06(void){
    HPS   hps;
    RECTL rclClip;
    GpiQueryClipBox(hps, &rclClip);
} /* fncCLIP06 */
```

Color and Mix Attributes

This chapter describes color and mix attributes and their use in OS/2 applications. The following topics are related to information in this chapter:

- Presentation spaces and device contexts
- Line and arc primitives
- Marker primitives
- Area primitives
- Character string primitives
- Bit maps and images primitives

About Color and Mix Attributes

Color and mix are two attributes of graphics primitives. They can be specified by an application in a number of ways, and are specified in the ...BUNDLE data structures associated with the five graphics primitives and in bit maps. Bit maps and some of the primitives have both foreground and background color attributes. For example, a character string primitive has a foreground color attribute that specifies the color of the character and a background color attribute that specifies the color of the character cell surrounding the character.

The mix attribute controls how each primitive is combined with the existing drawing. Among other things, it affects the color that results when primitives of different colors overlap. Primitives with foreground and background color attributes also have foreground and background mix attributes.

Color Implementation

Understanding how colors are implemented can assist in understanding how the color and mix attributes work.

The *pel* is the smallest element of the display screen that can be addressed. For monochrome displays, pels are either turned off or turned on. For color displays, each pel contains a red, green, and blue section, each of which is called a *phosphor* .

The display has color guns of red, blue, and green light that illuminate the phosphors in a single pel. By switching these color guns on and off in different combinations, eight standard colors can be produced. By varying the intensities of the color guns, additional colors can be produced. The following table shows the eight standard colors that can be generated from the three color guns.

Pel appears...	Red	Green	Blue
Red	ON	OFF	OFF
Green	OFF	ON	OFF
Blue	OFF	OFF	ON
White	ON	ON	ON
Black	OFF	OFF	OFF
Cyan (Turquoise)	OFF	ON	ON
Pink	ON	OFF	ON
Yellow	ON	ON	OFF

Each pel is described internally by a number of bits of storage. In a monochrome display, only one bit per pel is required, and that bit is either on or off. In an eight-color system, three bits per pel are necessary. Each of those three bits records the on or off setting of one color gun.

To be able to control the intensity of a color and obtain more than eight colors, more than three bits per pel are needed. For example, six bits per pel provide 32 different combinations.

The wider the range of available colors, the more bits per pel required for each color. This storage issue is resolved by keeping a wide choice of colors but restricting the number of colors available at any given time. Applications define the colors that they want to use in a */logical color table* . Selecting a color defined in the logical color table produces the nearest available color in the hardware palette.

RGB Color Encoding

The red, green, and blue (RGB) components of a color are stored in either an RGB or RGB2 data structure, or as a long integer (32-bit)

value. The color fields in the RGB2 structure and in the long integer follow the same rules.

If stored as a long integer, the RGB value has the first 8 bits reserved for a flag value and the remaining 24 bits reserved for color intensity. The flag byte must be set to 0. Each of the last three bytes specifies a color intensity, in the range 0 through 255, for a single primary color.

If a byte contains 0, the corresponding primary color is not present. As the value in the byte increases, the intensity of the primary color increases. For example, if the byte contains 128, the primary color is pale; if the byte contains 255, the primary color is as intense as the device permits.

The RGB value is determined by the following equation:

```
RGB Value = (R * 65536) + (G * 256) + B

Where:
    R is the red intensity
    G is the green intensity
    B is the blue intensity
```

If all three bytes are set to 0, the resulting color is black; if they all are set to 255, the resulting color is white. The RGB value associated with each of the standard eight colors is in the following table.

Color	RGB value
Black	0x00000000
Red	0x00FF0000
Green	0x0000FF00
Blue	0x000000FF
Pink	0x00FF00FF
Cyan	0x0000FFFF
Yellow	0x00FFFF00
White	0x00FFFFFF

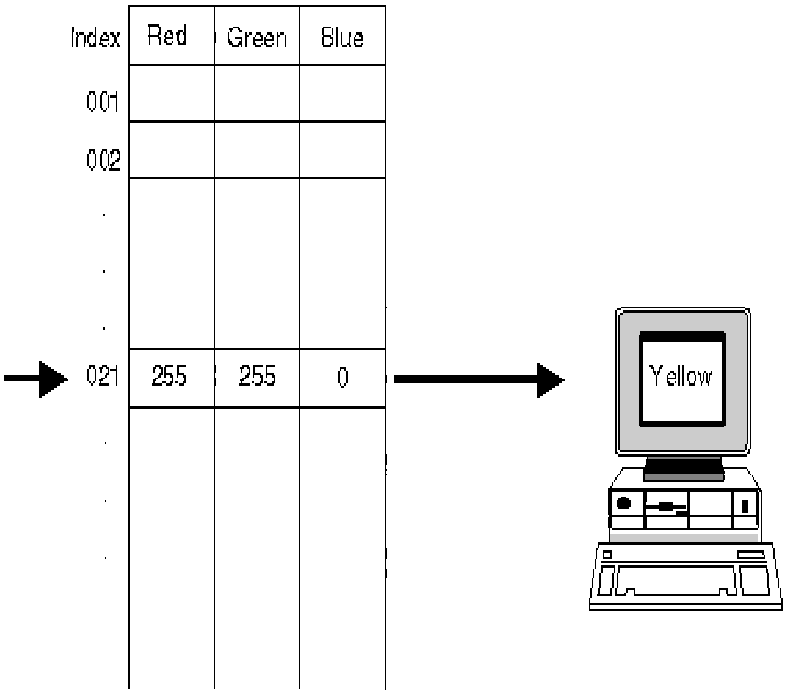
Color Tables

A color table is an array of RGB values. There are two kinds of color tables: logical and physical. The logical color table is a list of colors specific to a presentation space. An application typically uses a logical color table to define colors specific to that application.

The physical color table specifies colors the device can generate currently. These device colors are shared by every application on the system. Because some display adapters cannot generate every possible device color at the same time, the physical color table can be a subset of the full range of possible device colors. The operating system maps the RGB values specified in the logical color table to device colors in the physical color table.

Logical Color Table

A logical color table contains a variable number of entries, each of which describes a different RGB (Red, Green, Blue) combination that produces a color. The principle of the color table is illustrated in the following figure.



Logical Color Table

This simplified example demonstrates that to produce yellow on the computer screen, red and green are mixed in equal intensities, and no blue is used at all. In this example, yellow is addressed in the color table by index 21. Notice that this is not the same index number used in the default logical color table, which indicates that this color table has been changed by the application.

A logical color table is stored in a presentation space and is specific to that presentation space. A logical color table enables applications to specify colors as indexes rather than explicit RGB values.

The colors displayed are likely to vary from one output device to another so the definitions in the color table can be fine-tuned to get the best results. Any color can be made more or less intense by altering its definition in the logical color table.

Default Logical Color Table

The PM provides a default logical color table, which defines the colors and the indexes that retrieve them, as shown in the following table.

Default Logical Color Table

Color Index	Index Number	Effect
CLR_FALSE	-5	All bits are set to 0.
CLR_TRUE	-4	All bits are set to 1.
CLR_DEFAULT	-3	Default value
CLR_WHITE	-2	White
CLR_BLACK	-1	Black
CLR_BACKGROUND	0	Natural background color for the device
CLR_BLUE	1	Blue

CLR_RED	2	Red
CLR_PINK	3	Pink
CLR_GREEN	4	Green
CLR_CYAN	5	Cyan
CLR_YELLOW	6	Yellow
CLR_NEUTRAL	7	Neutral - The contrasting color to CLR_BACKGROUND
CLR_DARKGRAY	8	Dark gray
CLR_DARKBLUE	9	Dark blue
CLR_DARKRED	10	Dark red
CLR_DARKPINK	11	Dark pink
CLR_DARKGREEN	12	Dark green
CLR_DARKCYAN	13	Dark cyan
CLR_BROWN	14	Brown
CLR_PALEGRAY	15	Pale gray

Note: Entries after CLR_PALEGRAY have device-dependent defaults.

PM maps the color index name to the index number (shown in the second column of the previous table), then uses that number to address the appropriate color.

Device-Independent Color Indexing

Three of the index names provide a level of device independence in choosing colors: CLR_DEFAULT, CLR_BACKGROUND, and CLR_NEUTRAL. These indexes enable an application to select colors according to their purpose, and thus build device independence into your applications. The purpose of a color does not vary from one device to another, although the actual color used to implement that purpose might. The following table describes these indexes and the purpose of each:

Index	Purpose
CLR_BACKGROUND	The natural background color for the device. This is the color of the paper on a printer, and the window background color (white, by default) on a display.
CLR_NEUTRAL	The contrast color to CLR_BACKGROUND. This is usually black on a printer, and the default window text color (black, by default) on a display.
CLR_DEFAULT	Unless redefined, this has the same effect as CLR_NEUTRAL.

The colors produced by CLR_DEFAULT, CLR_BACKGROUND, and CLR_NEUTRAL in the default logical color table depend on the output device. For example, CLR_NEUTRAL could produce black on a device with a white background or white on a device with a black background.

Defining a Logical Color Table

To change the values in the default logical color table (that is, to load a new logical color table), use GpiCreateLogColorTable. Using this function, an application can do the following:

- Replace part or all of the default color table.
- Add color definitions to the default color table.
- Reset the logical color table to its default values.

There are two methods of making changes:

- To add to a table or change some of its contents, rather than replace the table completely, supply an array of color indexes and their associated RGB values.

- To load a consecutive sequence of index values, supply only an array of RGB values without index values.

Color Tables in Index Mode

The default logical color table is defined as an array of color indexes and their associated RGB values. A table in this format is in *index mode* .

To alter a logical color table in index mode, specify either LCOLF_INDRGB or LCOLF_CONSECRGB as the format value in GpiCreateLogColorTable. If an application uses the color index names to address the contents of a loaded color table in index mode, the color identified by the associated index number is retrieved. For example, CLR_NEUTRAL produces the color addressed by index number 7. If CLR_NEUTRAL must be blue, the application must define the color table so that index number 7 addresses the RGB definition of the appropriate shade of blue.

If an application uses the format LCOLF_INDRGB when calling GpiCreateLogColorTable, the application supplies an array of index and RGB pairs to update the table. These values do not have to be consecutive. An application can change the contents of an existing table and add entries to the end of the table.

If an application calls GpiCreateLogColorTable using the LCOLF_CONSECRGB format, the application supplies an array of RGB values and a starting index. When using this format, the RGB values must be consecutive. You cannot use a single call to GpiCreateLogColorTable to redefine CLR_BLUE and CLR_PINK without also specifying an RGB value for CLR_RED. As with LCOLF_INDRGB, this format can be used to change the contents of the existing table and to add entries to the end of the table.

When an application calls GpiSetAttrs to put a value into the color field of a graphics primitive attribute structure, the value is an index into the logical color table. When the operating system draws the primitive, it uses this index to determine the RGB value specified in the logical color table by the application. It then searches the physical color table for the color closest to this RGB value. The operating system then draws the primitive, using the color from the physical color table. How closely the drawn colors match the colors in the logical color table depends on the device colors in the physical color table.

Color Tables in RGB Mode

The color values specified with GpiSetAttrs and GpiSetColor also can be specified directly as RGB values rather than as indexes. To enable this, the logical color table must be switched into RGB mode by calling GpiCreateLogColorTable, specifying a format of LCOLF_RGB. (No color array is passed.)

If the color table is in RGB mode, it can be switched back to index mode and reset to its original default values by specifying either LCOLF_INGRGB or LCOLF_CONSECRGB in GpiCreateLogColorTable.

Querying the Available Colors

Several query functions are provided for applications to get information about the current logical and physical color tables before defining a

logical color table.

Applications use GpiQueryColorData to determine whether the default logical color table is in effect. If it is not, the following information about the loaded logical color table is returned:

- Format of the current logical color table
- Smallest and largest color indexes (if the table is in index mode) supported in this color table. The smallest color index is always 0, and the largest color index is never less than 15, because deleting entries from the default logical color table is not permitted.

GpiQueryRealColors returns the RGB values of each of the distinct colors defined in the physical color table of the current device. It also can return the index in the current logical color table that references each of the colors in the physical table.

To determine the available colors nearest to a specified color on a particular device, call GpiQueryNearestColor, which accepts as input the RGB value of the desired color. The function returns the RGB value of the nearest available color in the physical color table of the associated device.

Applications can use GpiQueryRGBColor to determine the RGB value of a particular color. GpiQueryRGBColor accepts as input the index to the logical color table entry of the color in question. Output from this function is the RGB value that the index would reference in the physical color table. If a logical color table is loaded in RGB mode, GpiQueryRGBColor returns the same results as GpiQueryNearestColor.

Conversely, to determine the index value that references a given RGB color (or the closest match to that color) in the physical color table, an application uses GpiQueryColorIndex. The application can determine which colors are in the current logical color table by calling GpiQueryLogColorTable. To determine which colors are in the physical color table, call GpiQueryRealColors.

Physical Color Table

Each output device has its own *physical color table* , which is organized like a logical color table. The physical color table contains the RGB color definitions of the distinct colors that the device can produce, while a logical color table contains the definitions of the colors as chosen by an application. When an application draws on an output device, the PM maps the index value that addresses the current color in the logical color table to the index that retrieves the closest match for the current color in the physical color table.

Because this substitution occurs when the graphics are drawn, the presentation space can be associated with a number of different device contexts without invalidating the logical color definitions. For example, an application can create a picture on the screen using a wide range of colors, then direct that drawing to an eight-pen vector plotter. The drawing is reproduced without having to be re-created. Although the picture, as drawn by the plotter, does not have the variety of colors displayed on the screen, the substitution process selects the nearest match for each color on the screen from the eight available colors on the plotter.

Palette Manager

Applications can use GpiCreatePalette to change the physical color palette by creating a new palette. This function is used if the application has specific color needs and must ensure that the color is part of the physical color palette. GpiCreatePalette also enables an application to prevent *dithering* of the 16 default colors by setting the flag parameter to LCOL_PURECOLOR. If the full 256 colors in the palette are needed, the application can set the flag to LCOL_OVERRIDE_DEFAULT_COLORS. It is recommended that the palette manager functions be used only when really necessary because the operating system cannot guarantee consistent colors in other windows and the ability to change the physical palette is device dependent.

After the palette is created, an application can set or change any of the values in the palette by calling GpiSetPaletteEntries. To delete the palette, applications use GpiDeletePalette, passing the handle of the palette to be deleted. (The handle is returned by GpiCreatePalette when the palette is created.)

GpiQueryPalette enables applications to determine the palette currently selected for a given presentation space. A palette can be used by more than one presentation space at the same time. The operating system maintains a count of the number of presentation spaces using a specific palette. Complete information on the current palette can be accessed by GpiQueryPaletteInfo.

Palettes also can be shared by using a palette handle.

When coding an application, color table functions and color palette functions must not be mixed. The application can call DevQueryCaps to determine whether the hardware supports palettes, then call the appropriate functions.

Realizing a Color Palette

WinRealizePalette maps the colors requested by the application into the color palette for the system. When the application's window is activated, the palette changes are transferred, or *realized*, into the physical palette for the system.

When a palette is realized, there might not be enough empty palette entries to accommodate the additional palette changes (a maximum of 256 entries). For example, it is possible that some of the colors changed in the physical palette are being used by another application. In this situation, a WM_REALIZEPALETTE message is posted to all the applications running on the desktop.

On receiving the WM_REALIZEPALETTE message, applications using palette manager functions must repaint their screens. The original application colors are mapped to the closest matching color in the new palette. Applications that do not use palette manager functions normally perform default processing within their applications causing a repaint of their windows with the closest match from the palette. If there are no changes, just additions, to the physical palette, no message is sent.

Note: The palette manager maps a window's colors to the closest available value in the palette when the physical palette has to be changed, but this does not guarantee that the color will be a close match to the original color used.

As the focus changes from window to window, the physical palette changes according to the activated window. Notification messages are sent as necessary to other applications.

If the physical color values in the palette have to change to accommodate the number of palette entries passed from the application when a palette is realized, the number of altered entries in the physical palette is returned by WinRealizePalette.

Color palette realization is available only to systems that have a minimum of 256 colors.

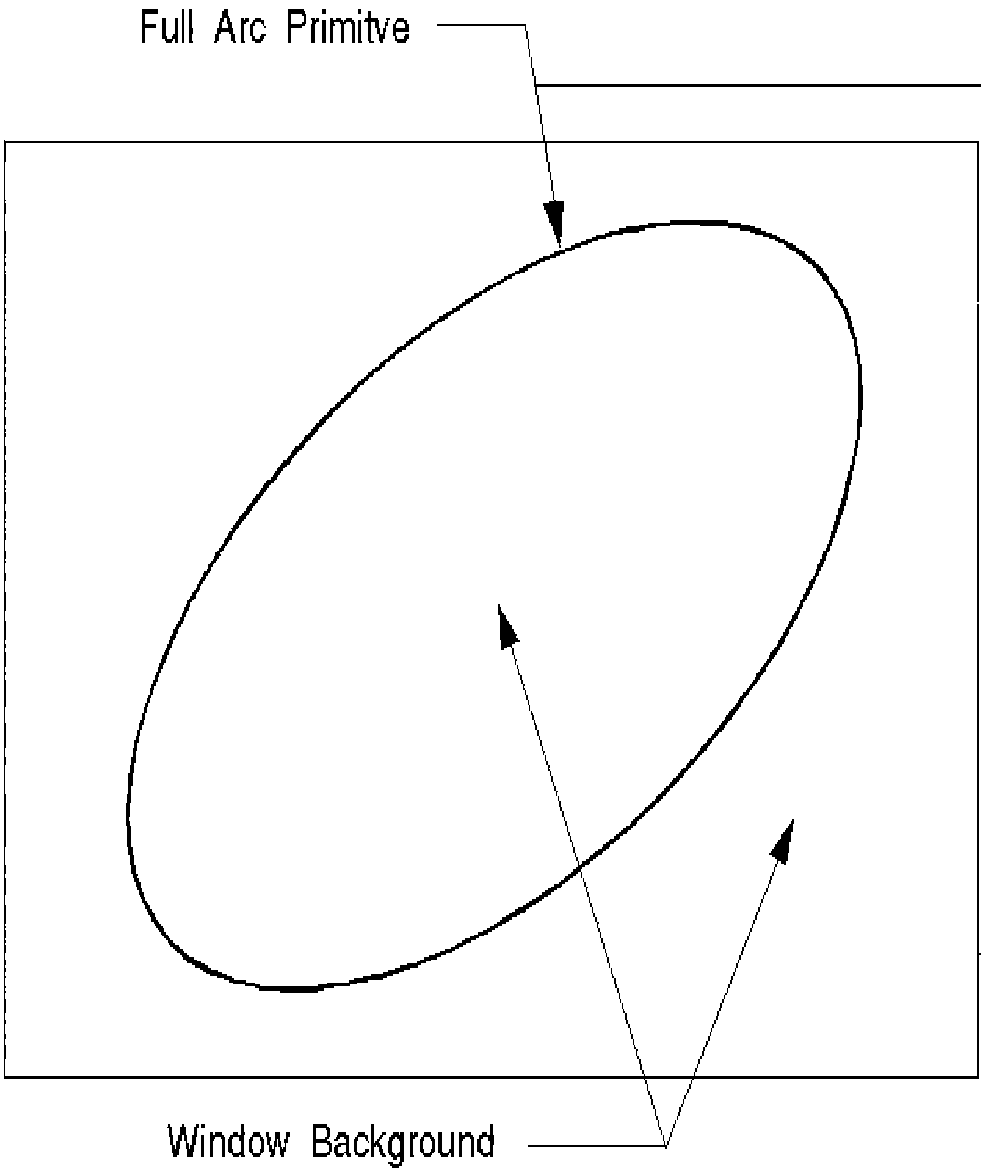
Color Attribute

The PM graphic interface uses a variety of colors. These colors are referred to as the *system colors* , and they are defined in the system color table, which is separate from an application's logical color table. To find out the RGB values of the system colors, call WinQuerySysColor.

The colors of graphic primitives are specified separately from the system colors. Every primitive has a foreground color, and some also have a background color.

Primitive Foreground

The foreground of a primitive is the primitive itself. For example, the foreground of a full arc primitive is the full arc, as shown in the following figure.



Foreground of a Primitive

The full arc primitive is drawn in a different color from the window background.

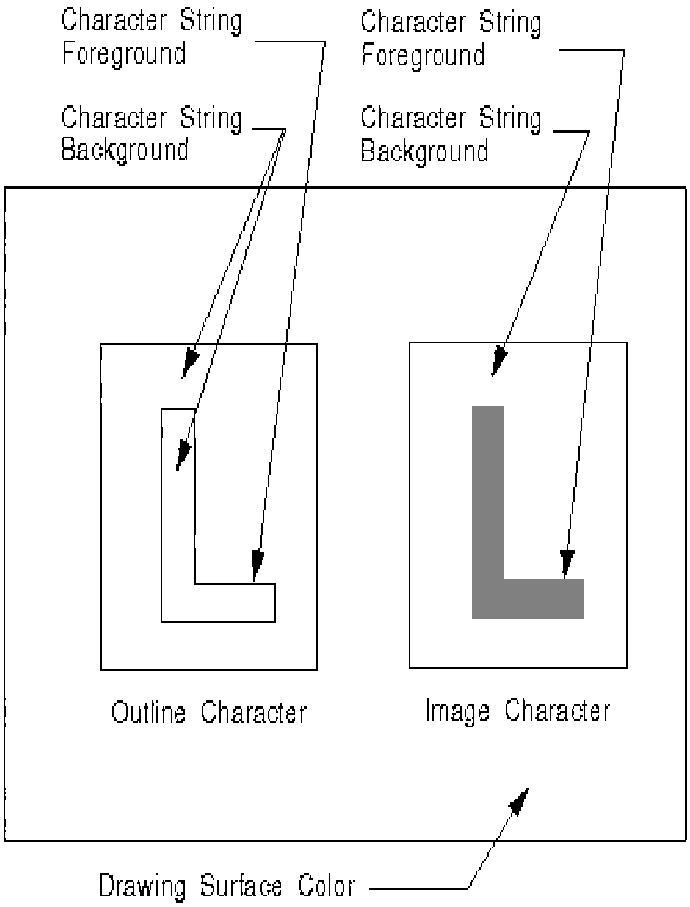
By default, the foreground color of all primitives is the color addressed by the index CLR_DEFAULT. In the default color table, this produces black on a graphics display. If the application replaces the default logical color table, CLR_DEFAULT produces the color addressed by index number 7 (CLR_NEUTRAL).

Primitive Background

The following primitives have a background:

- Areas
- Character strings
- Images
- Markers

The background of any character or marker primitive, whether the primitive is from an image or an outline font, is the entire character or marker box. First the background is drawn, then the foreground is drawn on top of it. Similarly, the background of an area primitive is the entire area to be filled. The background of an image primitive, however, is that part of the primitive in which the pels are not set. The following figure shows the background of a primitive.



Background of a Primitive

The background of this character-string primitive is the entire character box.

The index to the default background color is CLR_BACKGROUND, which provides a background color appropriate for the device. On a printer, CLR_BACKGROUND is the color of the paper. On a display screen, CLR_BACKGROUND is the default window background color.

Changing the Foreground and Background Colors of Primitives

To change the current foreground color, use GpiSetColor. As input, the application supplies either the index to the required color in the current logical color table or the RGB value of the color, depending on the mode of the table. Color indexes higher than those supplied in the default color table must be loaded explicitly before they can be used.

It is possible to specify one of the system colors (for example, SYSCLR_ACTIVEBORDER) as the current foreground (or background) color of a primitive. The color appears as defined in the system color table, and the logical color table is not used.

The specified color becomes the current color, and the foreground of any primitive drawn subsequently is drawn in that color. The current foreground color for a particular primitive type is set by calling GpiSetAttrs. For example, if an application sets the current foreground color to CLR_RED by calling GpiSetColor, and sets the current foreground color for marker primitives to CLR_CYAN by calling GpiSetAttrs, all

subsequent markers are cyan, and all other primitives are red.

To change the current primitive background color, call GpiSetBackColor. To ensure that this color is different from the background color of the output area (so that the entire primitive background is visible), the current window background color must not be the same as the current primitive background color. Also, select an appropriate background mix attribute.

GpiQueryBackColor returns the current background color setting for a character primitive. To learn the current foreground color setting, use GpiQueryColor. GpiQueryAttrs can be used to determine the current foreground and background color values for a single primitive type.

Color Output and Mix Attributes

A *mix attribute* is a bitwise operation on the color indexes in a device's logical color table. Mix attributes enable colors to be specified in relation to other colors. When an application draws a graphics primitive, the operating system uses a mix attribute to determine the color that appears on the output device.

For example, instead of specifying that a line should be black and, therefore, invisible if the background also is black, an application can use a mix attribute of FM_XOR to specify that the line always should be drawn the inverse of the background color.

Suppose an application has set the **lColor** field in the LINEBUNDLE structure to CLR_RED and the **usMixMode** field in the same structure to FM_OR. The current color of the drawing surface is CLR_GREEN. To determine the color of a line, the operating system performs a bitwise OR operation on the indexes of these colors, as follows:

```
 0010  (the default index for red)
 0100  (the default index for green)
------
 0110  (result of bitwise OR)
```

In this case, the result is 6-the index for yellow. This means that even though the application specified CLR_RED in the LINEBUNDLE structure, a yellow line appears when the application calls any of the functions that draw a line primitive.

Mix Attribute

The mix attribute determines how each primitive an application draws is combined with any existing drawing. In color applications, the mix attribute determines the color that results when one primitive is drawn on top of another. There are two forms of the mix attribute: *foreground mix* and *background mix*.

The foreground mix attribute governs how the foreground of a primitive is combined with the existing drawing, and it applies to all primitive types. The background mix attribute governs how the background of a primitive is combined with the existing drawing, and it is applicable only to those primitives that have a background. Primitives that can be affected by the background mix attribute are areas, character strings, images, and markers. The primitive attribute data structures contain fields for both foreground and background color and mix attributes.

There are 17 foreground mix attributes. For each mix attribute, the indexes of the foreground and current drawing-surface colors are combined by using one of the bitwise operators. The available foreground mix settings are listed in the following table.

Foreground Mix Attributes

Mix Attribute	Effect	Description
FM_DEFAULT	Default	Default foreground mix attribute (overpaint).
FM_OR	OR	Index value of the final color is determined by a bitwise OR operation on the index of the foreground color and the index of the color of the drawing surface.
FM_OVERPAINT	Overpaint	Index value of the final color is that of the foreground color. This is the default foreground mix attribute.

FM_XOR	Exclusive-OR (XOR)	Index value of the final color is determined by a bitwise XOR operation on the index of the foreground color and the index of the color of the drawing surface.
FM_LEAVEALONE	Leave-alone (Invisible)	Index value of the final color is that of the index of the color of the drawing surface.
FM_AND	AND	Index value of the final color is determined by a bitwise AND operation on the index of the foreground color and the index of the color of the drawing surface.
FM_SUBTRACT	(Inverse Source) AND Destination	Index value of the final color is determined by inverting the index of the foreground color and performing a bitwise AND operation on this value and the index of the color of the drawing surface.
FM_MASKSRCNOT	Source AND (Inverse Destination)	Index value of the final color is determined by inverting the index value of the drawing-surface color and performing a bitwise AND operation on this value and the index value of the foreground color.
FM_ZERO	All zeros	RGB value of the final color's is always 0x00000000.
FM_NOTMERGESRC	Inverse (Source OR Destination)	Index value of the final color is always the inverse of the FM_OR result.
FM_NOTXORSRC	Inverse (Source XOR Destination)	Index value of the final color is always the inverse of the FM_XOR result.
FM_INVERT	Inverse (Destination)	Index value of the final color is always the inverse of the index of the color of the drawing surface.
FM_MERGESRCNOT	Source OR (Inverse Destination)	Index value of the final color is determined by performing a bitwise OR operation on the index of the foreground color and the inverse of the index of the color of the drawing surface.
FM_NOTCOPYSRC	Inverse (Source)	Index value of the final color is the inverse of the index of the foreground color.
FM_MERGENOTSRC	(Inverse Source) OR Destination	Index value of the final color is determined by performing a bitwise AND operation on the index of the drawing surface's color and the inverse of the index of the foreground color.
FM_NOTMASKSRC	Inverse (Source AND Destination)	Index value of the final color is the inverse of the FM_AND result.
FM_ONE	All 1's.	RGB value of the final color is always 0x00FFFFFF.

There are five background mix attributes. For each mix attribute, the index value for the background color and the current drawing-surface color (in the device's physical color table) are combined using one of the bitwise operators.

The RGB values are those from the *physical* color table. When the result of the mix attribute's bitwise operation defines a color different from that of both the drawing surface and the drawing attribute, the resulting index accesses an RGB color in the physical table. The color, therefore, is unpredictable unless the logical color table has been realized (using the palette manager).

The first five of the foreground mix attributes also are available as background mix attributes. The background mix attributes are listed in the following table.

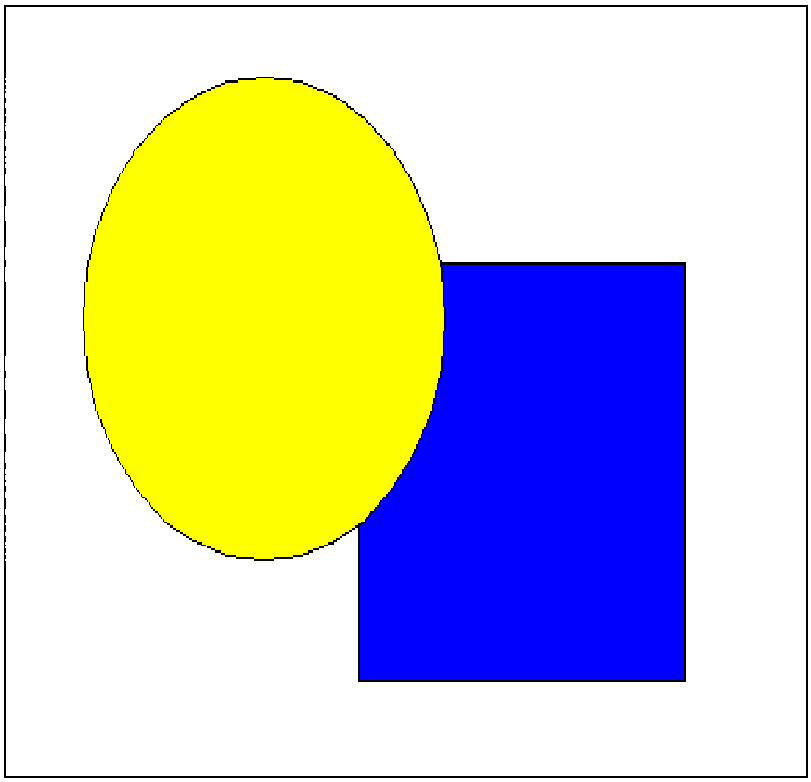
Background Mix Attributes

Mix Attribute	Effect	Description
BM_DEFAULT	Default	Default background mix attribute (Leave-alone).
BM_OR	OR	Index value of the final color is determined by a bitwise OR operation on the index of the background color and the index of the color of the drawing surface.
BM_OVERPAINT	Overpaint	Index value of the final color is that of the background color.
BM_XOR	Exclusive-OR (XOR)	Index value of the final color is determined by a bitwise XOR operation on the index of the background color and the index of the color of the drawing surface.
BM_LEAVEALONE	Leave-alone (Invisible)	Index value of the final color is that of the drawing-surface color.

The most frequently used foreground mix attributes are FM_OVERPAINT, which is the default value, FM_OR, and FM_XOR. The most frequently used background mix attributes are BM_LEAVEALONE, which is the default value, and BM_OVERPAINT.

Overpaint Mix Attribute

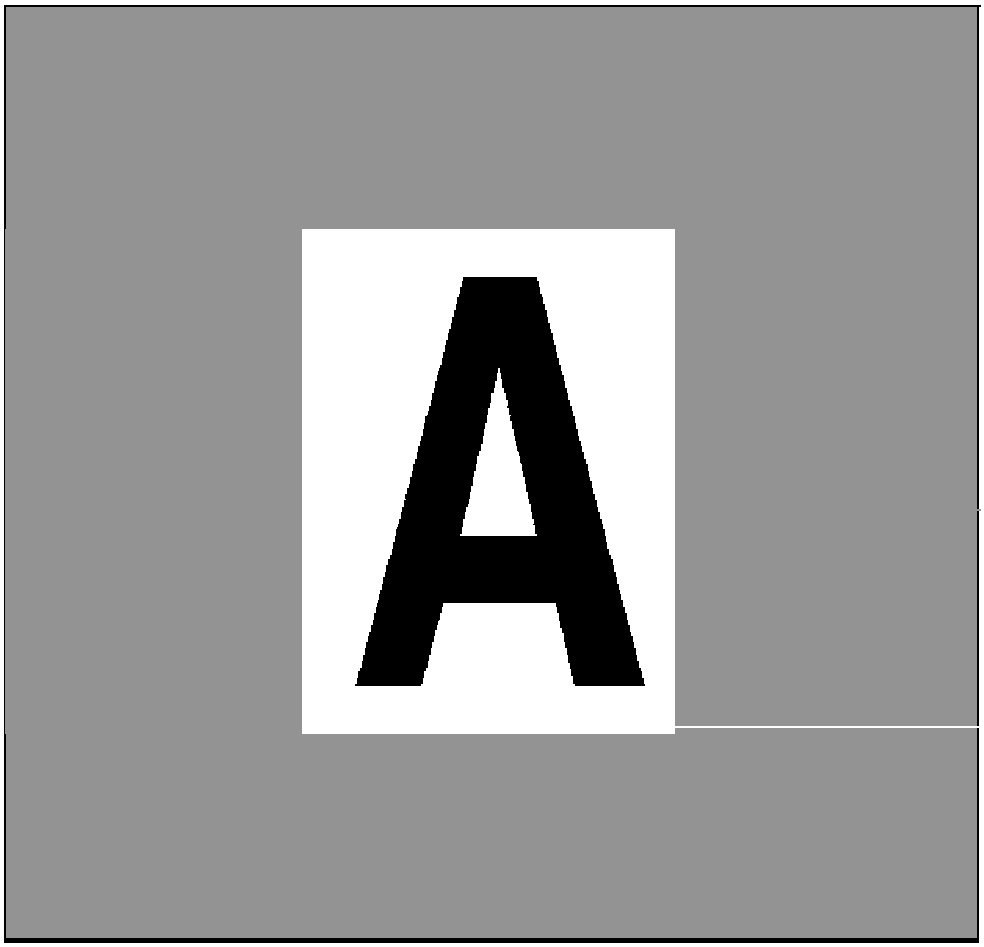
When using FM_OVERPAINT, the foreground of the primitive replaces any existing drawing in the same area of the presentation page. If the existing drawing is yellow, for example, and the new drawing is red, the drawing is red at the points of overlap. (This is the default foreground mix attribute.) Because one color is replacing another and no color mixing is being performed, the effects of the overpaint mix attribute are entirely predictable. This is shown in the following figure.



Overpaint Foreground Mix Attribute

The circle is drawn on top of the square. At the points of overlap, the output is the color of the circle.

When using BM_OVERPAINT, the primitive background replaces the existing drawing. drawing, as shown in the following figure.

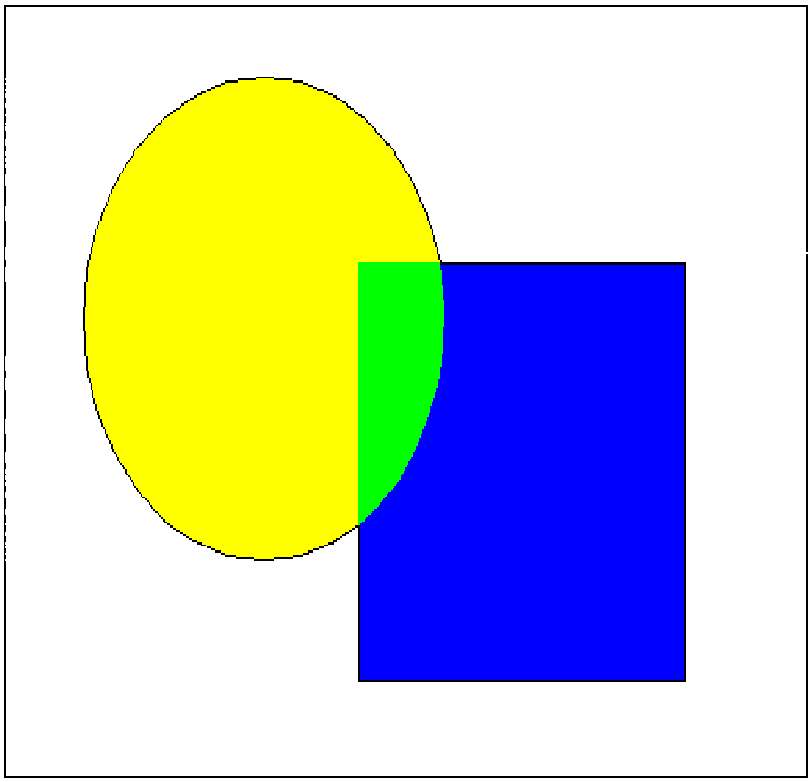


Overpaint Background Mix Attribute

Using BM_OVERPAINT, the background of the primitive is apparent only if it is drawn in a different color from the output-area background. Notice that, in this example, the foreground mix attribute is FM_OVERPAINT.

OR Mix Attribute

When using FM_OR, the foreground of the new primitive is merged with the existing drawing at the points of overlap. This is effected by ORing the indexes of the overlapping colors to produce a third color. The resulting color is unpredictable if the logical color table has not been realized (using the palette manager). The OR mix attribute is useful for making the common points of two graphics distinct from the points belonging to one of the graphics only, as shown in the following figure.



OR Mix Attribute

The circle is drawn on top of the square. At the points of overlap, indexes are OR'd to produce a new index referencing a new RGB color in the physical color table.

When using the BM_OR attribute, the background of the primitive is merged with the existing drawing according to the same rules that apply to the FM_OR attribute.

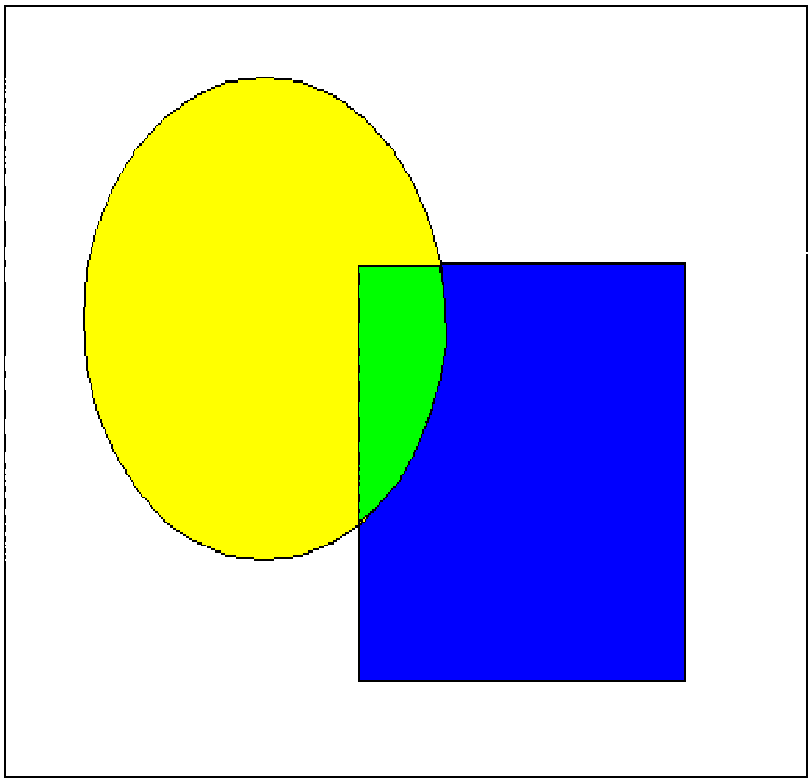
Exclusive-OR (XOR) Mix Attribute

The FM_XOR attribute enables objects to be drawn in such a way that they can be removed easily by simply drawing them a second time using the FM_XOR attribute. The FM_XOR attribute is available on display devices only and is useful for graphics animation when an application must move an individual graphic and completely restore the graphics that it originally overlapped. Typically, an application would do the following:

1. Draw the graphics object using the FM_XOR attribute.
2. Calculate the object's next position.
3. Draw the object again in its current position, still using the FM_XOR attribute. This effectively erases it from its current position without destroying the graphics with which it overlaps.
4. Draw the object in its new position using the FM_XOR mix attribute.

For retained graphics, this sequence can be automated to some extent by defining specific graphics segments as dynamic. Dynamic graphics always are drawn using the FM_XOR attribute, regardless of the current mix attributes.

The effects of the FM_XOR attribute are shown in the following figure.



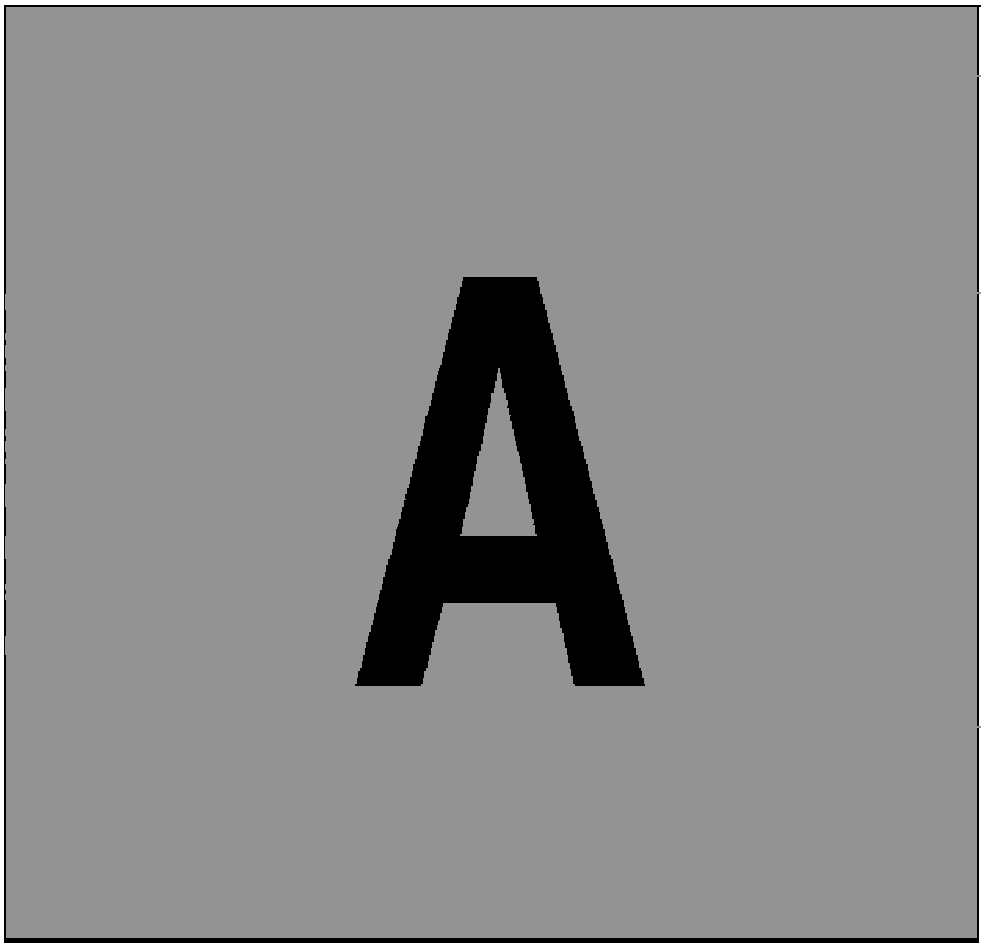
Exclusive-OR (XOR) Mix Attribute

The circle is drawn on top of the square. At the points of overlap, graphics are not drawn if the two overlapping figures are an identical color.

When using the BM_XOR attribute, the background of the primitive is merged with the existing picture according to the same rules that apply to the FM_XOR attribute.

Leave-Alone Mix Attribute

The leave-alone mix attribute most often is used as a background mix attribute. When using the FM_LEAVEALONE attribute, the foreground of the primitive is not drawn. When using the BM_LEAVEALONE attribute, the background of the primitive is not drawn, as shown in the following figure.



Leave-Alone Background Mix Attribute The leave-alone background mix attribute is the background mix attribute that most often is used for character strings and marker primitives. It is generally used with a foreground mix value of FM_OVERPAINT.

Specifying Foreground and Background Mix Attributes

Specify the current foreground mix value for all primitives using GpiSetMix. To learn the current foreground mix setting, use GpiQueryMix, which returns the current foreground mix setting for a character string primitive. GpiQueryAttrs can be used to determine the current foreground and background mix values for a single primitive type.

To specify a background mix value for all primitives, use GpiSetBackMix. GpiQueryBackMix returns the current background mix value for a character-string primitive. To specify the current foreground and background mix values for a single primitive type, call GpiSetAttrs.

Note: Not all devices support all of the background and foreground mix attributes described. When a device does not support the mix attribute chosen by an application, the default mix attribute is used; no error condition is raised. DevQueryCaps can be used to determine whether a mix attribute is supported on a specific device.

Color on Advanced Display Devices

Some devices can display simultaneously a fixed number of colors (typically 156), chosen from a much larger number of colors (often more than 256000). An application can use the palette manager functions to take advantage of the extra capabilities of these devices. These functions enable an application to change the colors in a device's physical color table and the displayed colors rapidly without explicitly

redrawing the screen.

An application can use the CAPS_ADDITIONAL_GRAPHICS option of DevQueryCaps to determine whether the display device supports palette functions.

Dithering

If an application requests a color not available in the physical color table, the operating system can approximate the color by a process called *dithering*. For example, if the physical color table does not contain a light green color but does contain a yellow and a green, the operating system can create what appears to be light green by mixing yellow pels and green pels. Dithering is a variation on the way red, green, and blue color guns illuminate the phosphors in a single pel to produce a color that is not pure red, green, or blue (for example, yellow).

The dithering process takes advantage of the way the human eye interprets color. If every other pel is set to one color, and all the intermediate pels to a different color, together they produce the effect of a third color at normal viewing distances.

The checkerboard effect is just one of the ways in which dithering can be implemented. Dithering works only when producing a solid mass of color, such as an area fill pattern. It does not have the desired effect on line primitives.

Dithering is especially important on monochrome devices. By combining various combinations of black pels with white pels, the operating system can create numerous shades of gray.

To use only the pure colors defined in the physical color table, that is, to prevent color dithering, set LCOL_PURECOLOR in GpiCreateLogColorTable. When LCOL_PURECOLOR is set, the nearest available color to the one selected is used.

Considerations When Using Monochrome Displays

When a graphic primitive that is drawn in color is displayed on a monochrome display, the operating system maps the colors that the application uses to the colors supported by the monochrome display.

Drawing Color Graphics on Monochrome Devices

When mapping color graphics to a monochrome device (screen, printer, or bit map), the monochrome device has a *reset* color and a *contrast* color. The reset color is the color GpiErase clears to and is either black or white. If the reset color is black, the contrast color is white, and if the reset color is white, the contrast color is black.

To determine whether the reset color is to be black or white, the color retrieved by the index CLR_BACKGROUND from the current logical color table is examined. If a logical color table has not been loaded, CLR_BACKGROUND is either the paper color on a printer or SYSCLR_WINDOW on a display. If a logical color table has been loaded, CLR_BACKGROUND is the color addressed by color index 0 on any device.

If the color retrieved by CLR_BACKGROUND is either white or black, that color becomes the reset color. For example, if an application is drawing to a screen, has not loaded a color table, and the system colors have not been altered, the background color is white, because SYSCLR_WINDOW produces a white background by default. In this instance, the contrast color is black.

If the color retrieved by CLR_BACKGROUND is neither white nor black, the color translates to whichever of black and white it is nearest to. As a rule, dark colors produce a black background, and pale colors produce a white background. For example, if an application is using a loaded color table, or if the color retrieved by SYSCLR_WINDOW has been altered (either interactively or by WinSetSysColors), CLR_BACKGROUND could be dark gray. In this instance, the reset color would be black, and the contrast color would be white.

When the reset color has been established, the PM applies the following rules when mapping color graphics to a monochrome device:

- Any color graphics drawn in CLR_BACKGROUND, and any graphics defined in the actual reset color (which is either black or white), are drawn in the reset color. Any other graphics are drawn in the contrast color.
- The index CLR_WHITE produces white, and the index CLR_BLACK produces black, regardless of whether the reset color is black or white.

- If no color table is loaded and CLR_DEFAULT or CLR_NEUTRAL are used as the foreground color, they produce the contrast color. If they are used as the background color, they produce the reset color.
- If an application calls GpiQueryNearestColor for a monochrome device, one of the following occurs:
 - If the color supplied on input is the reset color, the reset color is returned.
 - If the color supplied on input is not the reset color, the contrast color is returned.

Drawing Color Area Fill Patterns on Monochrome Devices

An area primitive is drawn according to the current foreground and background mix attributes and in the current area foreground and area background colors.

When an application draws a monochrome pattern on a color device, the bits of the pattern set to 1 translate to the current area foreground color, and the 0 bits translate to the current area background color. When the application draws a color pattern on a monochrome device, and if the current pattern is anything other than PATSYM_DEFAULT or PATSYM_SOLID from the default pattern set, the color closest to white is translated into 1 bits. For example, if a pattern of diagonal lines is being drawn in which the foreground color is red and the background color is cyan, the cyan is translated to white (1 bits) because cyan is closer than red is to white. Red, therefore is translated to black (0 bits). The effect of translating this color pattern to a monochrome surface is summarized as follows:

```
Pattern      As 1s and     Color       Monochrome
                0s        Surface       Surface

\   \        10001000     RcccRccc     01110111
 \   \       01000100     cRcccRcc     10111011
  \   \      00100010     ccRcccRc     11011101
   \   \     00010001     cccRcccR     11101110
\   \        10001000     RcccRccc     01110111
 \   \       01000100     cRcccRcc     10111011
  \   \      00100010     ccRcccRc     11011101
   \   \     00010001     cccRcccR     11101110
```

The original pattern of 1's and 0's is used, however, when deciding which part of the pattern is the background and which part is the foreground. Thus, if the background mix attribute is BM_LEAVEALONE, the following occurs:

```
Pattern      As 1s and     Color      Monochrome
                0s        Surface      Surface

\   \        10001000     R...R...     0...0...
 \   \       01000100     .R...R..     .0...0..
  \   \      00100010     ..R...R.     ..0...0.
   \   \     00010001     ...R...R     ...0...0
\   \        10001000     R...R...     0...0...
 \   \       01000100     .R...R..     .0...0..
  \   \      00100010     ..R...R.     ..0...0.
   \   \     00010001     ...R...R     ...0...0
```

The 1 bits on the monochrome surface still are interpreted as the background of the primitive and are not drawn when the BM_LEAVEALONE attribute is specified.

When a bit map is used as an area fill pattern, any bit drawn in the current area background color is set to 0, and all other bits are set to 1 on a monochrome surface. Thus, if the current area background color is blue, all blue bits in the bit map are set to 0, and all other bits are set to 1. The 0 bits constitute the background of the primitive.

If the pattern is solid (PATSYM_DEFAULT or PATSYM_SOLID in the supplied pattern set), the following occurs:

- If color dithering is switched off, and the application is drawing a color pattern to a color surface, the color nearest the color specified is used.
- If color dithering is switched on, and the application is drawing a color pattern to a color surface, a combination of colors can be used to achieve the effect of the requested color. For example, if the application chooses pink on a surface where pink is not available, a combination of red and white pels can be used to achieve the effect of the color.
- If color dithering is switched on, and the application is drawing a color pattern to a monochrome surface, sufficient pels are set to

suggest the intensity of the requested color.

Dithering can be enabled and disabled using LCOL_PURECOLOR in GpiCreateLogColorTable.

Using Color and Mix Attributes

The color- and mix-attribute functions can be used to perform the following tasks:

- Create a logical color table
- Determine the format and the starting and ending index values of the current logical color table
- Determine the index value for an entry in the logical color table that is the closest match to an RGB value
- Determine the RGB value associated with a particular entry in the logical color table
- Determine and set the current foreground and background colors
- Determine and set the current foreground and background mix attributes

Creating a Logical Color Table

To create a logical color table, the application creates an array of RGB values that replace the existing logical color table, then calls GpiCreateLogColorTable, using the LCOL_RESET and LCOLF_CONSECRGB flags. The following figure demonstrates this process.

```
#define INCL_GPILOGCOLORTABLE
#include <os2.h>

void fncCOLR01(void){
    HPS hps;
    LONG alTable[] = {
        0xFFFFFF,        /* White */
        0xFF88FF,
        0xFF8800,
        0xFF8888,
        0xFF0088,
        0x880088,
        0x008888,
        0x00FF88,
        0x00F800,
        0x008800,
        0x000088,
        0x0000F8,
        0x0800F8,
        0x888888,
        0x080808,
        0x000000 };      /* Black */

    GpiCreateLogColorTable(hps,
       LCOL_RESET,                     /* Start with the default      */
       LCOLF_CONSECRGB,                /* Consecutive RGB values      */
       0,                              /* Starting index in table     */
       sizeof(alTable) / sizeof(LONG),  /* Number of elements in table */
       alTable);
} /* fncCOLR01 */
```

Determining the Color-Table Format and Index Values

To determine the format and the starting and ending index values of the current logical color table, applications call GpiQueryColorData. The following figure is an example of using GpiQueryColorData to determine whether the default logical color table is loaded. If so, the code fragment loads a new table.

```
#define INCL_GPILOGCOLORTABLE
#include <os2.h>

void fncCOLR02(void){
    HPS hps;
    LONG        aClrData[3];
    LONG alTable[16];

    GpiQueryColorData(hps, 3, aClrData);

    if (aClrData[QCD_LCT_FORMAT] == LCOLF_DEFAULT)
        GpiCreateLogColorTable(hps,
           LCOL_RESET,                     /* Start with the default      */
           LCOLF_CONSECRGB,                /* Consecutive RGB values      */
           0,                              /* Starting index in table     */
           sizeof(alTable) / sizeof(LONG),  /* Number of elements in table */
           alTable);
} /* fncCOLR02 */
```

Determining the Index Value of an RGB Value

Applications call GpiQueryColorIndex to find the closest match in a logical color table to an RGB value. This function finds the closest match for this RGB value in the physical color table, then finds the color in the logical color table closest to the color in the physical color table. The function returns the index value of that entry in the logical color table. The followiing figure is an example of how to determine which index value in the logical color table matches the RGB value for pink (0x00FF00FF), then uses that index entry to set the foreground color to pink for each of the primitive attributes.

```
#define INCL_GPILOGCOLORTABLE
#include <os2.h>

void fncCOLR03(void){
    LONG lIndex;                            /* Logical-color-table index */
    HPS hps;

    lIndex = GpiQueryColorIndex(hps, LCOLOPT_REALIZED, 0x00FF00FF);

    if ((lIndex >= 0) && (lIndex <= 15))     /* Check for valid index     */
        GpiSetColor(hps, lIndex);
} /* fncCOLR03 */
```

Setting the Primitive Color Attributes

To set the color attributes for one type of graphics primitive, an application uses GpiSetAttrs. To set the color attributes for each type of primitive in a presentation space, use GpiSetColor and GpiSetBackColor. The following figure shows how to use GpiSetAttrs to set the color

attribute of line primitives to dark gray.

```
#define INCL_GPIPRIMITIVES
#include <os2.h>

void fncCOLR04(void){
    LINEBUNDLE lbnd;     /* Line-primitive attribute structure */
    HPS hps;

    lbnd.lColor = CLR_DARKGRAY;

    GpiSetAttrs(hps, PRIM_LINE, LBB_COLOR, 0, &lbnd);
} /* fncCOLR04 */
```

The following figure is an example of how to use GpiSetColor to set the foreground color attribute to dark gray in all of the primitives.

```
#include <os2.h>

void fncCOLR05(void){
    HPS hps;

    GpiSetColor(hps, CLR_DARKGRAY);
} /* fncCOLR05 */
```

Creating a Palette

To create a palette, an application must first call DevQueryCaps with the CAPS_ADDITIONAL_GRAPHICS option to determine whether the device supports palette functions. Next the application creates an array of RGB values or an array of RGB2 structures, then calls GpiCreatePalette. Next the applications calls GpiSelectPalette to select the palette for the presentation space and calls WinRealizePalette to map the RGB values to device colors for subsequent drawing. When the application is finished drawing, it calls GpiSelectPalette to disassociate the presentation space and the palette. Then it deletes the palette by calling GpiDeletePalette.

The following figure demonstrates these steps.

```
#define INCL_GPILOGCOLORTABLE
#define INCL_GPIBITMAPS
#include <os2.h>
void fncCOLR06(void){
    COLOR clrCurrent;
    HPAL hpal;
    HDC hdc;
    HPS hps;
    HAB hab;
    HWND hwnd;
    POINTL aptl[2], ptl;
    LONG cSimulColors, lPalSupport;
    SHORT j;
    RGB2 *prgb2ColorData;

    /* Determine how many colors the device can display at once.             */
    DevQueryCaps(hdc, CAPS_COLORS, 1, &cSimulColors);

    /* Determine if the device supports palette manager functions.           */
    DevQueryCaps(hdc, CAPS_ADDITIONAL_GRAPHICS, 1, &lPalSupport);

    /* Allocate space for the array of RGB2 structures.                      */
    DosAllocMem((PPVOID)&prgb2ColorData, cSimulColors * sizeof(RGB2), fALLOC);

    /* Fill the array of RGB2 structures with as many different              */
    /*    shades of blue as the device will support.                         */

    clrCurrent = 0x000000FF;
    for (j = 0; j < cSimulColors; j++) {
```

```
        prgb2ColorData[j].bRed = 0;
        prgb2ColorData[j].bGreen = 0;
        prgb2ColorData[j].bBlue = clrCurrent;
        prgb2ColorData[j].fcOptions = 0;
        clrCurrent = clrCurrent > 0 ? --clrCurrent : 0x000000FF;
    }

    if (lPalSupport & CAPS_PALETTE_MANAGER) {
        hpal = GpiCreatePalette(hab,       /* Create palette                  */
               0L,
               LCOLF_CONSECRGB,            /* Format of color table entries   */
               cSimulColors,               /* Number of entries in table      */
               (PULONG) prgb2ColorData);   /* Pointer to color table          */
    }

    GpiSelectPalette(hps, hpal);
    WinRealizePalette(hwnd, hps);
    GpiSelectPalette(hps, NULLHANDLE);     /* Restore default physical colors */
    GpiDeletePalette(hpal);                /* Delete palette                  */
} /* fncCOLR06 */
```

Correlation

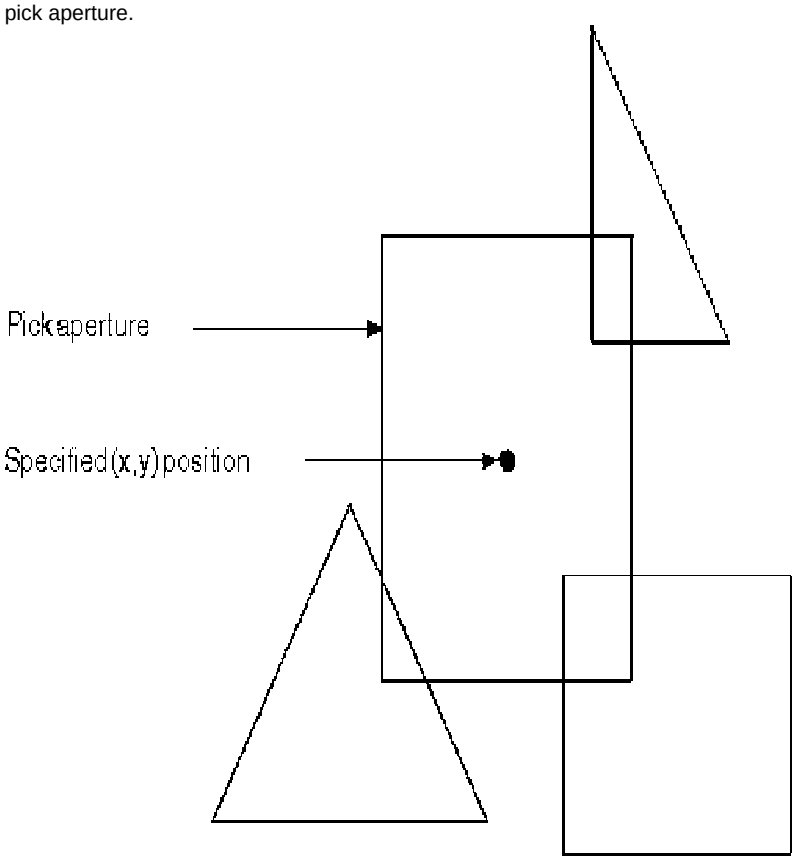
Correlation is the process of determining which primitives, if any, are contained in an area of interest. The process for nonretained graphics is very different from the process for retained graphics. The following topics are related to the information in this chapter:

- Presentation spaces
- Coordinate spaces and transformations
- Editing retained graphics and graphics segments

About Correlation

When you want to select an area of interest on the screen, you usually move the pointer to the applicable point and signal (by clicking a mouse, for example) that this is the object you want. This selection process most commonly is used for graphic applications. For example, your application could permit a user to select an object, then change its color. Correlation, however, can also be used in nongraphic applications. For example, your application could model the operation of a calculator and permit a user to select numbers for mathematical operations.

The *area of interest* is defined by the operating system as a small rectangle, centered on the (x,y) coordinate position, that has been sent to the application. The virtual rectangle the application generates is known as the *pick aperture* . The following figure shows an example of a pick aperture.



The Pick Aperture

The process of determining what lies in the pick aperture differs between nonretained and retained graphics. For nonretained graphics, the presentation space must be put in a state that supports correlation before the primitives are drawn. Then correlation is performed while the primitives are being drawn. For retained graphics, the segments that contain the graphics can be replayed, permitting correlation after the primitives are drawn.

Note: The necessity to correlate graphics can usually be predicted. Therefore you can plan for it when designing applications. However, this

does not mean that you have to create retained (rather than nonretained) graphic segments. The decision to retain graphics is dependent on several considerations of which correlation is only one.

Correlating Nonretained Graphics

For the purposes of correlation, *nonretained graphics* are those graphics that are being correlated *during* the drawing process. Nonretained graphics can exist in nonretained graphic segments or completely outside any segment structure. Primitives outside segments are detectable when the applicable draw control is set.

Nonretained graphics, inside a segment bracket, can be created in either *draw* or *draw-and-retain* modes. If created in draw-and-retain mode, a segment, at first, is considered nonretained while the primitives in the segment are being drawn; then it is considered retained. To be correlated, nonretained segments must have unique, nonzero identifiers, and must be defined as *detectable* . The primitives within these segments can be *tagged* just as primitives in retained segments are. However, the tags do not influence the correlation process for nonretained graphics.

To get correlation data from the drawing of nonretained graphics, three steps must be performed- after creation of the presentation space but before drawing the primitives:

1. Call GpiSetDrawControl to switch on the *correlation flag* (DCTL_CORRELATE, DCTL_ON).

2. Call GpiSetPickApertureSize, if necessary, to change the size of the pick aperture.

3. Call GpiSetPickAperturePosition, if necessary, to explicitly position the aperture. As input to this function, you provide the coordinate position on which the pick aperture is to be centered, using presentation page coordinates.

Correlation is performed for the following functions:

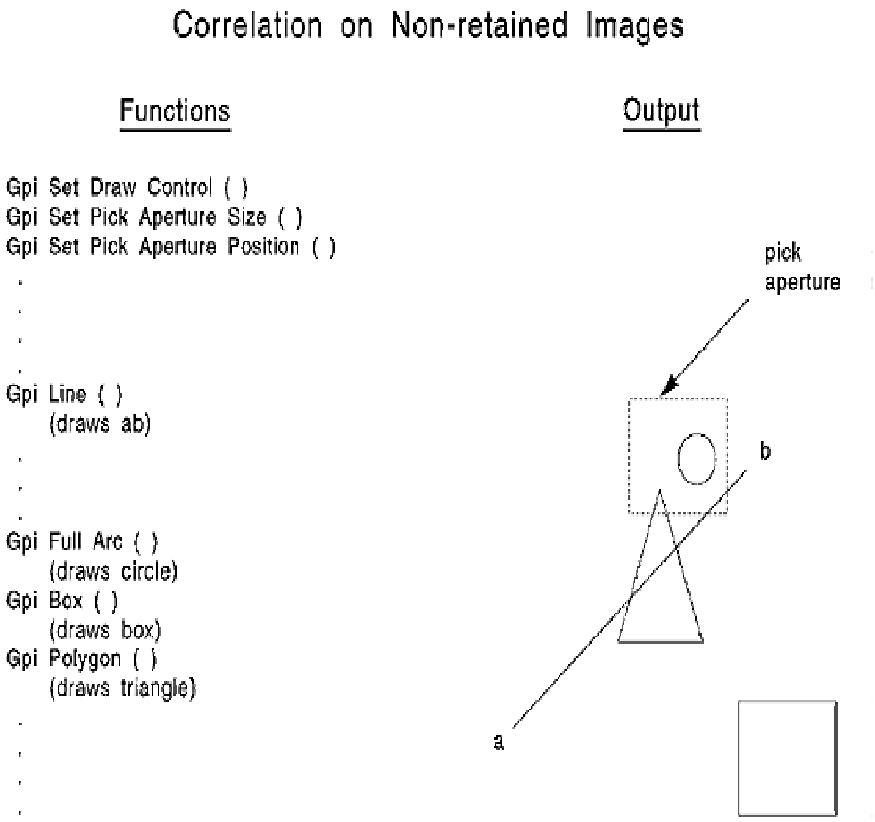
- Individual primitive-drawing requests, for example, GpiBox
- GpiPutData
- GpiElement
- GpiPlayMetaFile

Correlation is never performed for GpiErase

You detect correlation *hits* by examining the returned values from the GPI functions. If GpiLine, for example, draws a line that intersects the pick aperture, it returns a value of GPI_HITS to indicate a correlation hit. If the line does not intersect the pick aperture, GpiLine returns a value of GPI_OK, to indicate the successful drawing of a line without a correlation hit. GpiLine returns a value of GPI_ERROR if an error is detected.

If the line intersects the pick aperture, a correlation hit is returned even if the line style is LINETYPE_INVISIBLE. For other primitives, if the object is drawn in outline mode, a correlation hit is returned only if the pick aperture intersects the boundary. If the object is in fill mode, a correlation hit is returned if the pick aperture intersects or lies within the boundary.

The following figure is an example of primitives intersecting the pick aperture.



Correlating Nonretained Graphics

Each GPI function whose output intersects the pick aperture returns a hit (GPI_HIT). GpiBox, whose output does not intersect the pick aperture, returns GPI_OK.

Correlating Retained Graphics

The information in a retained graphic segment can be redrawn; therefore, setting up an environment for correlation is not necessary. Correlation on retained segments is processed through segment IDs and tags throughout the segments.

For any retained segment to be a candidate for correlation, the following must be done:

1. Call GpiOpenSegment with a unique, nonzero identifier.
2. Call GpiSetSegmentAttrs with the following settings:
 a. ATTR_DETECTABLE switched ON
 b. ATTR_DYNAMIC left OFF (default),
 unless the segment is unchained and, therefore, drawn non-dynamically
 c. ATTR_PROP_DETECTABLE left ON (default),
 in any segment that calls the candidate segment
3. Call GpiSetTag at appropriate locations within the segment.

Note: The preceding is the recommended method. However, an application still can receive correlation data about invisible segments.

Tagging Primitives within a Segment

GpiSetTag inserts a long integer value, called a *tag* , at the current element pointer position. The tag becomes a segment element, which serves as an identifier for the primitives that follow until the next tag is issued. Tags also are called *pick tags* or *pick identifiers* . Typically, applications assign tags only to elements that correspond to primitives. You can determine the value of the last tag assigned to an element using GpiQueryTag.

Note: Tags are used only in correlation. They are not used in the chaining or calling of segments, nor do they cause or modify output.

The long integer value of the tag is *0* by default. However, a 0 value makes the subsequent primitives undetectable. An application can use this effect to make some parts of a segment detectable and other parts of it non-detectable. If you know that this capability is unnecessary, you can change the default tag value using GpiSetDefTag.

The tag you specify becomes the current tag, and it applies to all subsequent primitives until you next call GpiSetTag. The tag can be considered one of the attributes of the primitive and therefore affected by the current attribute mode. In AM_PRESERVE mode, it is stored on the LIFO stack and can be recalled with GpiPop. Tags cannot be inserted between a GpiBeginArea and GpiEndArea area bracket. All primitives within an area have the same tag.

A tag value greater than 0 enables correlation on the subsequent primitives, but only if the segment ID is greater than 0 as well. The data returned from each correlation consists of a set of segment-tag pairs. The reason for this pairing is that a single segment can draw objects in several locations. By adding identifiers within the segment, an application has a more exact description of what has intersected the pick aperture.

For simple chained segments, each unique segment-tag pair within the pick aperture is known as a *hit* . If two or more primitives in the pick aperture have the same tag, they are considered a single hit. Hits for called segments differ slightly and are described in The lMaxDepth Input Parameter.

Correlation Functions for Retained Graphics

The PM has three different functions that enable you to correlate a uniquely identified retained segment.

Function	Description
GpiCorrelateSegment	Permits correlation on a single segment.
GpiCorrelateFrom	Permits correlation on a range of segments from a segment chain.
GpiCorrelateChain	Permits correlation on the entire segment chain.

For nonretained graphics, the correlation hits are returned by the actual drawing commands. For retained graphics, the correlation hits are returned by the three correlation functions listed above.

The size of the pick aperture is set using GpiSetPickApertureSize, just as with nonretained graphics. However, the coordinate position on which the pick aperture is centered usually is obtained from operator input, for example, from a WM_BUTTON1DOWN message, instead of from GpiSetPickAperturePosition.

After the graphics orders that create pictures are stored in retained segments, the pictures can be re-created by your application with the various GpiDraw... functions. Then the user can view the pictures, if the output is directed to a screen for example. After the user selects an area of interest, a GpiCorrelate... function redraws the picture internally to determine just what intersects the pick aperture. The user does not see the re-creation. A standard order of functions within your application would be:

- GpiDraw...
- GpiSetPickApertureSize
- GpiCorrelate...

The Correlation Input Parameters

There is only one segment chain per presentation space. Therefore GpiCorrelateChain needs the presentation space handle as input. The segment correlation functions need both the presentation space handle and the segment IDs.

All three correlation functions require the following as input:

- Correlation attribute type
- Maximum number of hits
- Number of segment-tag pairs to be returned for a single hit, called the *pair depth*

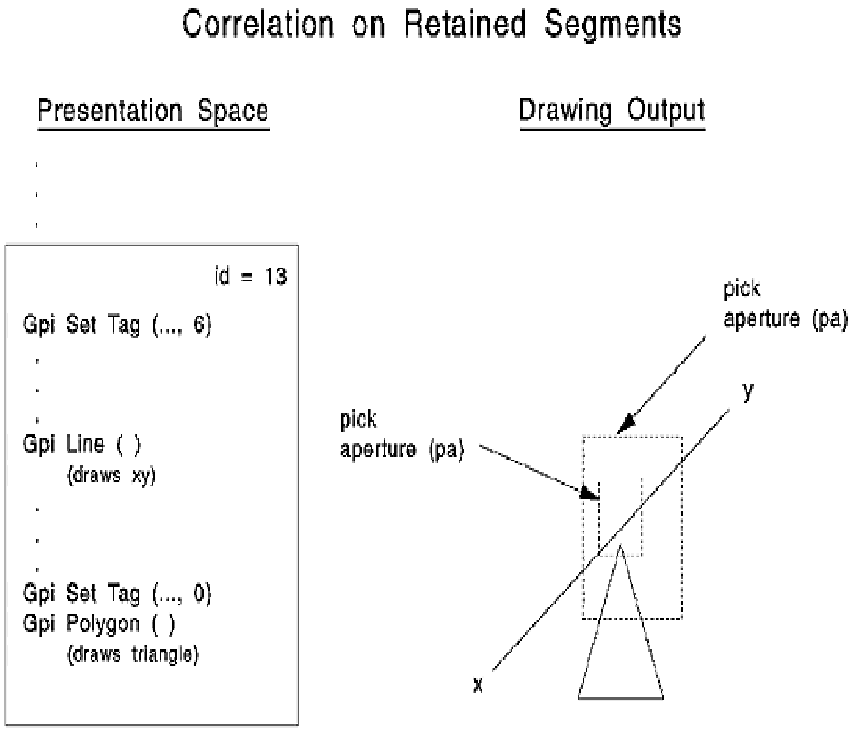
The lType Input Parameter

The two classifications of segments upon which you can request correlation are the following:

- Segments that have been defined as both detectable and visible
- All nonzero segments, regardless of their detectability and visibility attributes

The lMaxHits Input Parameter

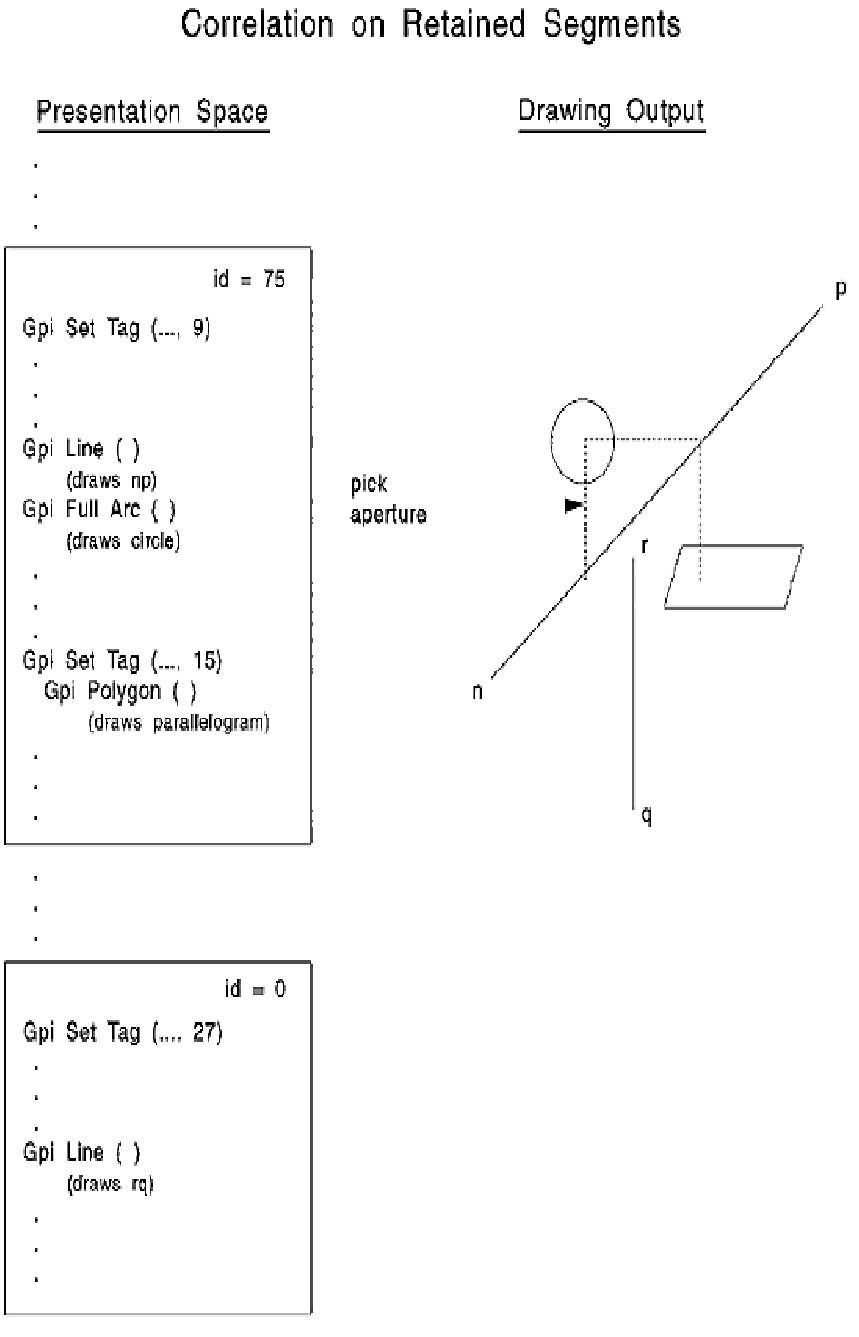
The correlation functions return the number of hits made on the retained segments. The following figure is an example of a line intersecting the pick aperture.



Retained Segment Correlation with One Hit

The intersection of a unique segment-identified and -tagged primitive with the pick aperture (pa) produces one hit. If the pick aperture size were increased, there still would be only 1 hit. The triangle produces no hit because its tag is 0.

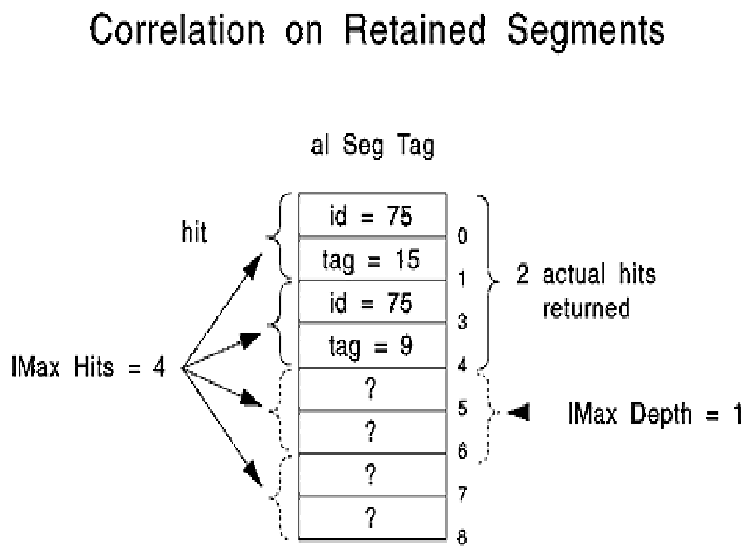
The following figure is an example of multiple primitives intersecting the pick aperture.



Retained Segment Correlation with Multiple Hits

Four separate primitives intersect the pick aperture; however, since two primitives share the same tag, and one primitive has a segment ID of 0, there are only two hits.

Your application can set a limit on the number of hits to return from a correlation function. The maximum-number-of-hits parameter influences the size of the array created to handle the segment-tag pair returned for each hit. By comparing the maximum number desired to the actual number of hits, your application can determine whether all hits are accounted for. The following figure shows an example of the alSegTag data structure that contains the segment-tag pairs for the previous figure.

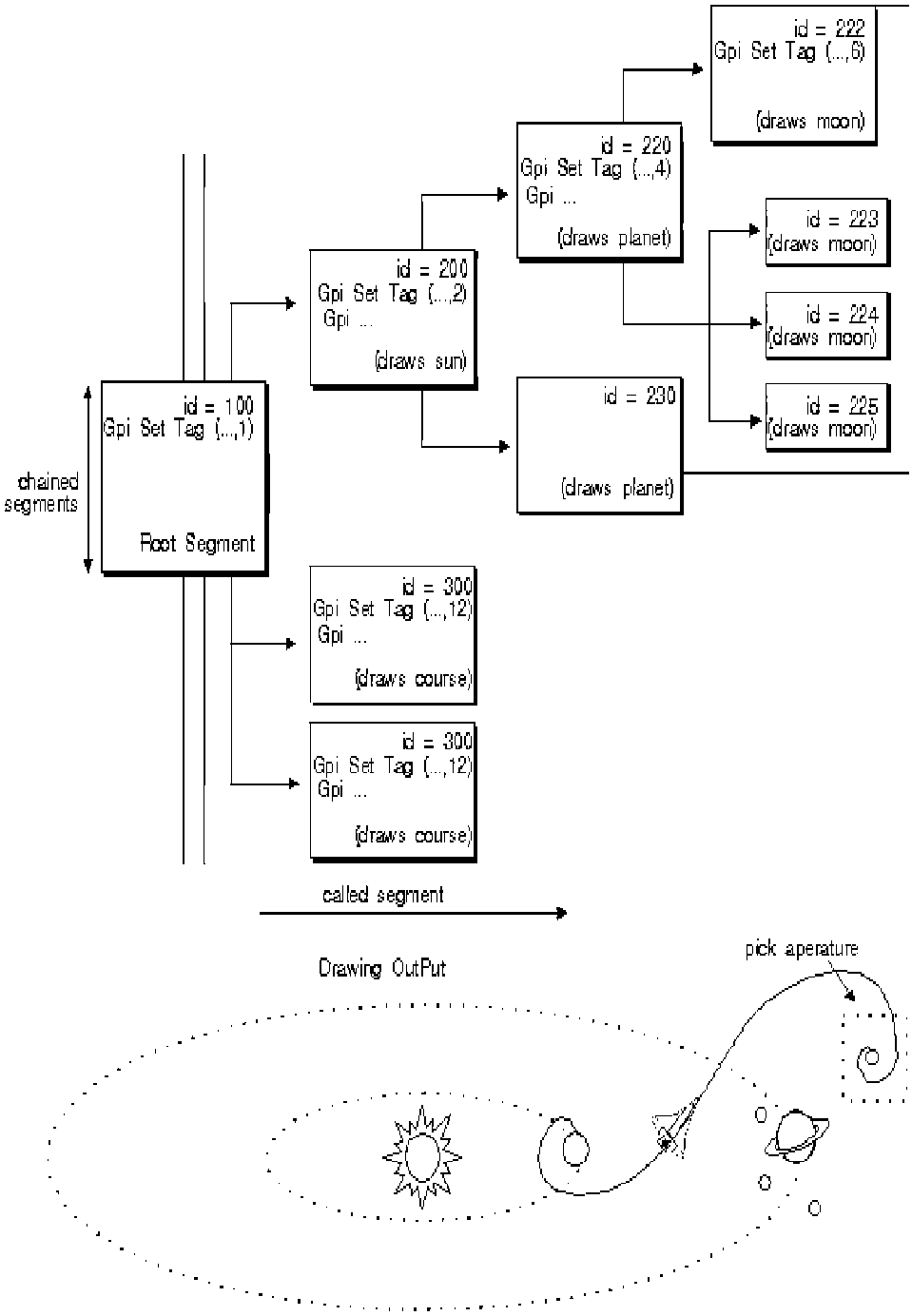


alSegTag Data Structure

As shown in the previous figure, the The identifier-tag pairs are returned to the application in the reverse order of their occurrence on the segment chain. That is, the highest priority segment is returned first. Therefore, the application can identify the topmost segment, which is the segment most likely to have been picked.

The lMaxDepth Input Parameter

When a called segment is picked, correlation data is returned also for all segments above it in the hierarchy, up to and including the root segment. The following figure is an example of a picture drawn from a complex segment chain with called segments.

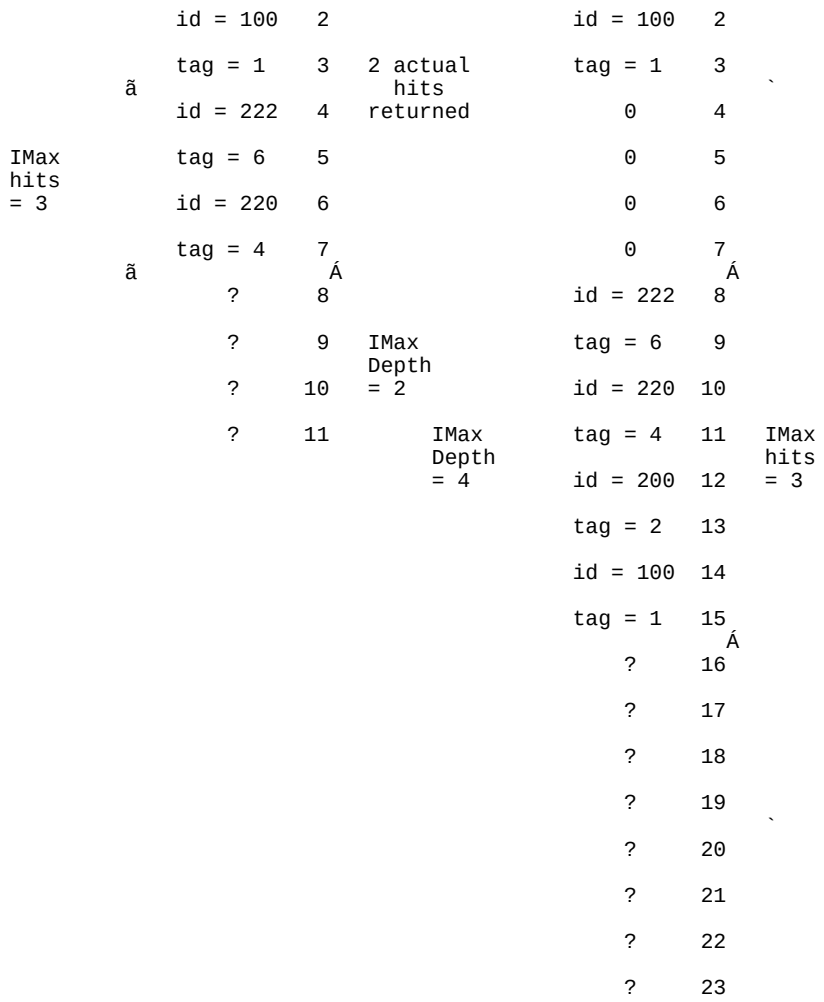


Multiple Hits from a Called Segment

Two separate items, called from different portions of the segment chain, intersect the pick aperture. Each has a unique segment identifier and tag, so there are two hits.

For called segments, the group of segment-tag pairs constitutes a single hit. You can limit the number of segment and tag pairs returned for each hit using the maximum depth parameter, just as you can limit the total number of hits returned to you using the maximum number of hits parameter. The following figure shows two examples of the alSegTag data structure from the previous figure, for two different lMaxDepth values.

```
id = 300   0                    id = 300   0
tag = 12   1                    tag = 12   1
```



alSegTag Data Structure for Different lMaxDepth Values

Unused segment-tag pairs for actual hits are set to 0 in the alSegTag array.

There are two major reasons for the PM to provide this capacity. The first is the consideration of application storage. If your application is a graphics package, for example, providing extensive design capabilities to an end user, the user's drawing may be very complex with 10 or more levels of segment calling. The data returned from a single hit could require an alSegTag array so large the data overruns the application storage you had reserved. By setting a maximum calling depth, your application can reserve the correct amount of storage.

The second consideration is the knowledge that your application's user is interested only in a certain level of calling depth. Many users will be interested only in the topmost called segment, because it usually is the segment containing the functions that performs the actual drawing.

Pick Aperture

Your application determines the size of the pick aperture using GpiQueryPickApertureSize, and determines the page space coordinates of the center of the aperture with GpiQueryPickAperturePosition.

Because your objective is to retrieve a single detectable object for each correlation operation, you have to obtain a balance between the number of detectable objects in a picture and the size of the pick aperture. The more detectable objects you define, the smaller the pick aperture must be to return a manageable amount of information to the application. However, the pick aperture usually is not a 1 pel-by-1 pel rectangle. A larger rectangle makes better sense because the pick aperture can be difficult to hit with a single pel; but a larger sampling area, for example 4 pels-by-4 pels, increases the probability of an intersection between a lighted pel and the pick aperture.

The pick aperture can be specified in any of the units available to the presentation space. However, the type of unit must be identical to the type specified for the presentation page. The default size of the pick aperture is square, with sides equal to the default character cell height.

Using Correlation

This section explains how to perform correlation on the graphics associated with a segment.

The following figure shows how to set the pick aperture size with GpiSetPickAperture and correlate the segment chain with GpiCorrelateChain, passing it the pick-aperture position as the third argument. The functions use the psizlPick and pptlPick arguments to set the aperture size and position the aperture center. The correlation is performed on visible and detectable segments only (PICKSEL_VISIBLE). GpiCorrelateChain copies any segment-tag pairs to the buffer pointed to by alSegTag and returns the count of hits detected.

```
#define INCL_GPICORRELATION
#include <os2.h>
LONG fncCORL01  (HPS hps, PSIZEL psizlPick, PPOINTL pptlPick,
                 LONG lMaxHits, LONG lMaxDepth, PLONG alSegTag)
    {
        LONG cHits;

        GpiSetPickApertureSize(hps, PICKAP_REC, psizlPick);
                                                /* Set the pick aperture. */

        cHits = GpiCorrelateChain(hps, PICKSEL_VISIBLE, pptlPick,
                    lMaxHits, lMaxDepth, alSegTag);
                                                /* Correlate the hits.    */

        return (cHits);                         /* Return count of hits.  */

} /* fncCORL01 */
```

Coordinate Spaces and Transformations

A *transformation* is an operation performed on a graphics object that changes the object in one of four ways: translation, rotation, scaling and shearing. Transformations enable an application to control the location, orientation, size, and shape of graphics output on any output device.

The transformation of graphics output can be conceptually divided into a series of distinct transformations applied from one logical stopping point to another. Coordinate spaces are used as a method of conceptualizing these logical stopping points. The coordinate spaces are concepts used to explain and manipulate the transformation process.

A graphics primitive in an intermediate coordinate space cannot be displayed on an output device and a transformation cannot be interrupted at one of the distinct stages. The entire series of transformation steps is applied all at once.

This chapter describes the transformation design process. The following topics are related to the information in this chapter:

- Presentation spaces and device contexts
- Segments and retained graphics
- Clipping

About Coordinate Spaces

Most of the GPI functions draw their output in a conceptual area called the *world coordinate space* . If you picture the presentation space as a blank canvas on which to draw, the world coordinate space is a Cartesian grid that provides a reference of scale for what is being drawn.

The components of a picture defined in a world coordinate space are often defined to a scale convenient only to that component. Applications also can define each component, or *subpicture* , starting at the origin (0,0). This enables applications to define the scale of a subpicture and the location of the subpicture separately. The ability to define all subpictures at the origin means there is not always 1-to-1 correspondence between a presentation space and a world coordinate space. Frequently a separate world coordinate space exists for every subpicture.

After subpictures are defined in the world coordinate space, they undergo a process called transformation and then appear on an output device. If an application has not specifically applied a transformation to its subpictures, by default the PM applies the *identity transformation* , which, in effect, makes no changes to the subpicture.

All graphics systems, not just the PM, use at least one coordinate space to generate output. The simplest graphics systems use a single coordinate space, whose points are the pels on the display. These are sometimes called single-space systems.

The PM creates multiple coordinate spaces and uses transformations to move subpictures between the coordinate spaces. The PM assembles the application's graphic output in a process that can use up to five coordinate spaces, known collectively as the *viewing* pipeline. The PM coordinate spaces are:

- World coordinate space
- Model space
- Page space
- Device space
- Media space.

For those who are more familiar with a graphics system that uses a single-coordinate space, the following table lists the differences to expect when using PM's multi-coordinate space.

Single-Coordinate Space Systems	PM Multi-Coordinate System
Distances and locations specified in pels.	Seven units of measurement, including pels, are available.
Because the shape, or aspect ratio, and size of pels can vary on different devices, the graphic primitives on one device might look different on another.	The output is device independent.
Coordinates entered must already be device coordinates.	Coordinates are made device-compatible through the PM.

Additionally, a single-space system requires that applications keep track of:

- The scaling between the subpictures. Often this is accomplished by forcing the definition of objects to the scale of the picture, instead of to the scale of the subpicture.
- The scaling between the creation space and output space. Often this is accomplished by forcing the definition of objects to the scale of the paper or window, instead of to the scale of the subpicture.
- All aspects of other transformations.

World Coordinate Space

The world coordinate space is where most drawing coordinates are specified. Each primitive in a world coordinate space is defined using the coordinate positions in the primitive's definition. For example, if an application draws a line from (8,4) to (20,75), the coordinate values (8,4) and (20,75) are world coordinates.

The world coordinate space is specified when the presentation space is created with GpiCreatePS. The PS_FORMAT option supports two sizes of world coordinate spaces: short and long. If the size is short (GPIF_SHORT), the world coordinate space is a rectangular Cartesian space with maximum coordinates of (32767, 32767) and minimum coordinates of (-32768, -32768). If the world coordinate space is short, it is the application's responsibility to ensure that the coordinates defining the subpicture fall within this range.

The second size, long (GPIF_LONG), is the default for the world coordinate space. Most GPI functions that direct their output to the world coordinate space, for example GpiLine, accept 32-bit integers as parameters. A 32-bit integer is equivalent to a rectangular Cartesian space with maximum coordinates of (134217727, 134217727) and minimum coordinates of (-134217728, -134217728).

World coordinates do not have to be within the range of the presentation page. A graphic image that requires a high degree of detail and precision might use most or all of the available coordinate range. A transformation is used to scale the graphic image down to an appropriate size. The actual presentation page size is unimportant in most cases. It is used with the page viewport to redefine the device transform if GpiSetPageViewport is called.

An application can use a *clip path* clipping area to define the part of the world space to place in the next coordinate space (the model space). A clip path is the only clipping region that can be nonrectangular. Its edges can include arcs, curves, and straight lines. The coordinates that define the dimensions and shape of a clip path are always world coordinates. (Clipping is further described in Clipping and Boundary Determination.)

In a world-coordinate space, there can be several graphic primitives. If, however, an application uses the DM_RETAIN drawing mode to store output in graphic segments, the operating system assigns a new world space to each segment. There is also a world space for the drawings outside of segments.

Model Space

Model space is the conceptual area where the separate components of a picture, defined in world space, are brought together. To assemble one or more primitives from world spaces to model space, the application specifies the transformations to occur. Then the PM applies them to each of the components. Model transformations convert world coordinates to model-space coordinates. For example, an octagon and the word STOP, defined individually in separate world-coordinate spaces, can be assembled into a stop sign in model space.

Graphics applications can have more than one model space. If there is more than one model space, the picture components are assembled in page space. If an application has each model space in a different segment, the model transforms are reset for each segment.

An application can use a *viewing limit* clipping area to define the part of a model space to place in the next coordinate space (the page space). A viewing limit is always rectangular, and the coordinates that define its location and dimensions are always model coordinates.

Page Space

The page space is where a complete picture is assembled for viewing on a display screen, or for printing or plotting on a piece of paper. Page coordinate units can be increments of an inch, a meter, pels, or some arbitrary value. An application uses GpiCreatePS to specify the units used for page coordinates.

If the application uses retained-drawing mode, the picture in page space contains parts of models (or pictures) from each of the model spaces that have not been clipped. The picture also contains any additional unclipped graphics-primitive output that the application generated in nonretained-drawing mode. If the application uses nonretained-drawing mode, the picture in page space contains all the graphics-primitive output that has not been clipped.

In the page space, an application can use a rectangular clipping area called a *graphics field* to define the part of the page space to place in the next coordinate space (the device space). The coordinates that define the location and dimensions of the graphics field are always page coordinates.

The page space contains the presentation page whose size and units are defined when creating the presentation space with GpiCreatePS. The presentation page is a rectangle in page space.

Device Space

The device space is the coordinate space in which a picture is drawn before it appears in a display screen window or on the printer or plotter.

Device space is defined in device-specific units. Depending on the page unit used, device-coordinate units can be pels, increments of an inch, increments of a meter, or arbitrary. For example, if the page unit is pels, the device-coordinate unit is pels; if the page units are arbitrary, the device-coordinate units are arbitrary.

Media Space

The media space is used only with windows. When an application draws a primitive at a specified location in a window, it is not actually drawing in the device space, as the device is the entire terminal screen.

Drawing in a window involves a shifting transformation which moves a drawing from the given (unitless) position, to position in the specified window.

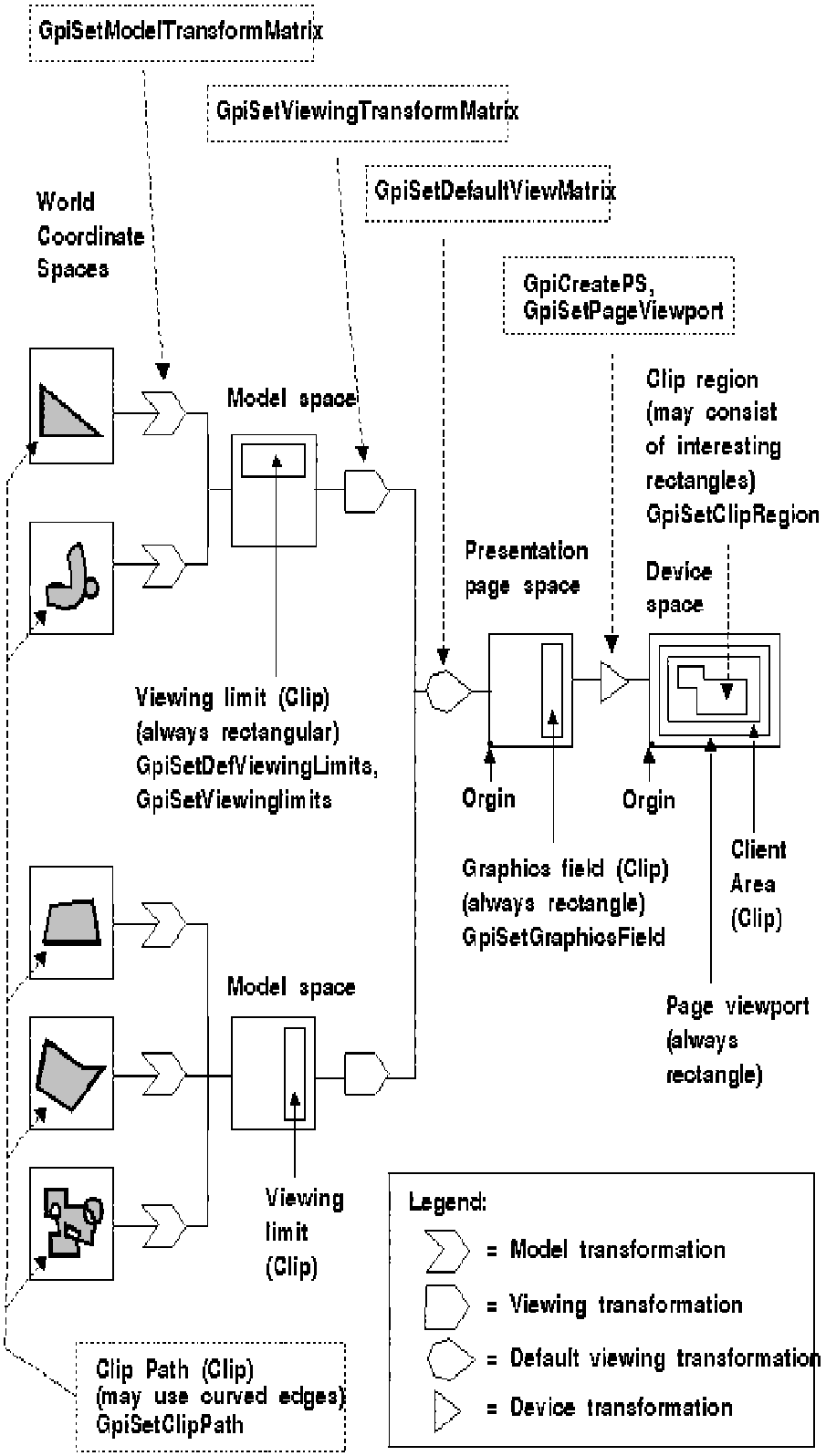
About Transformations

The coordinate spaces are connected by different transformation functions. In reality, the input coordinates of the world coordinate space are transformed directly into device coordinates in a single operation. The intermediate coordinate spaces exist only to provide a useful model to assist in the understanding of how to define the different transformation matrixes.

Transformation of graphic primitives occur when a transformation matrix is applied to those primitives. The individual transformations introduced here do not actually transform the primitives, but rather define portions of the transformation matrix. After all portions of the matrix are identified, the actual operation is performed in a single step.

The transformation functions manipulate graphics primitives as they move from one coordinate space to the next. Transformation functions usually begin with the letters *Gpi*. Most of the function names have a *transformation type* that identifies the entities on which it operates. For example, a model transformation is the transformation type that transforms graphics primitives between a world coordinate space and a model space. There is a 1-to-1 correspondence between these transformation types and the actual functions. The transformation matrix data structure is called MATRIXLF.

The following figure lists the sequence of coordinate spaces and the transformation types between the coordinate spaces. Internally, all transformations are combined, and the resulting values are held in the same format as the individual components. By default, there is no viewing window and no graphics field. The application can use the default page viewport. Clipping regions for each coordinate space are also shown.



Viewing Pipeline

Identity Transformation

The *identity transformation* is the default transformation between all coordinate spaces. The identity transformation, makes no change to the original coordinates of an object. This transformation is also referred to as the *unity transformation*, because of the mathematical matrix used to make this transformation.

The identity matrix looks like this:

```
1  0  0
0  1  0
0  0  1
```

(Transformations are accomplished with matrix multiplication; and 1, not 0, is the multiplication identity.)

If a transformation is not explicitly specified, the identity transformation is used to create that portion of the transformation matrix. Most transformations applied to a primitive can be in addition to, or instead of, the current transformation.

Moving through Coordinate Spaces Using the Identity Transformation

The effects of transformation can be minimized by defaulting to the identity transformation. This means that no deliberate action is requested by the application.

Graphic primitives are drawn in their own coordinate scale in world coordinate space. The primitives are then combined in model space. Neither the world coordinate space or the model space have "real world" units associated with them. This means that if an application draws a line on the x-axis from -300000 to +300000, the line exists as a line 600000 Cartesian units long in both world coordinate and model space.

Page space is the first of the coordinate spaces to be associated with units such as millimeters or inches. These units are set with the GpiCreatePS option PS_UNITS. If the application specifies the option PS_UNITS with the value PU_ARBITRARY, the page space remains unitless; units are applied when the output is drawn in device space.

As an example, assume that the units PU_LOMETRIC, (0.1 mm) have been chosen. Again, assuming the identity transformation between model and page space, the primitive appears in the page space as a line that extends

```
300000 * 0.1mm to the right of the origin
300000 * 0.1mm to the left of the origin
```

A presentation page, a rectangle in page space, has its size defined when a presentation space is created with GpiCreatePS. The presentation page format is defined with the GpiCreatePS option, PS_FORMAT. The three choices for that option are:

- GPIF_DEFAULT-32-bit integers or 2GB
- GPIF_LONG-32-bit integers
- GPIF_SHORT-16-bit integers or 32KB.

PS_FORMAT affects the number of units permitted along the coordinate axes of the presentation page. If the choice was GPIF_LONG or GPIF_DEFAULT, and, for example, PS_UNITS remains PU_LOMETRIC, the presentation page axes could range from 0 to 134217727 0.1mm. If the choice was GPIF_SHORT, the coordinate axes could range only from 0 to 32767 0.1mm. These limits are the maximum number of units, presentation pages are usually much smaller.

The presentation page does not affect the primitive in presentation space, it determines what parts of the primitive are visible. With no transformation to scale down the primitive from the model to the page space, the presentation page acts like a sheet of cardboard with a rectangular hole. Only the part of the page space behind the "hole" is visible in page space. The view of everything else is blocked by the "cardboard." Only what is visible in the page space is drawn to the next coordinate space.

Continuing with the example of the 600000 units line above, if GPIF_LONG (or GPIF_DEFAULT) is selected, the 600000 unit line could easily be viewed in the presentation page. If GPIF_SHORT is selected, the maximum presentation page size is 65534 * 0.1 mm units long. In the case, much of the 600000 unit line would not be drawn into the next coordinate space.

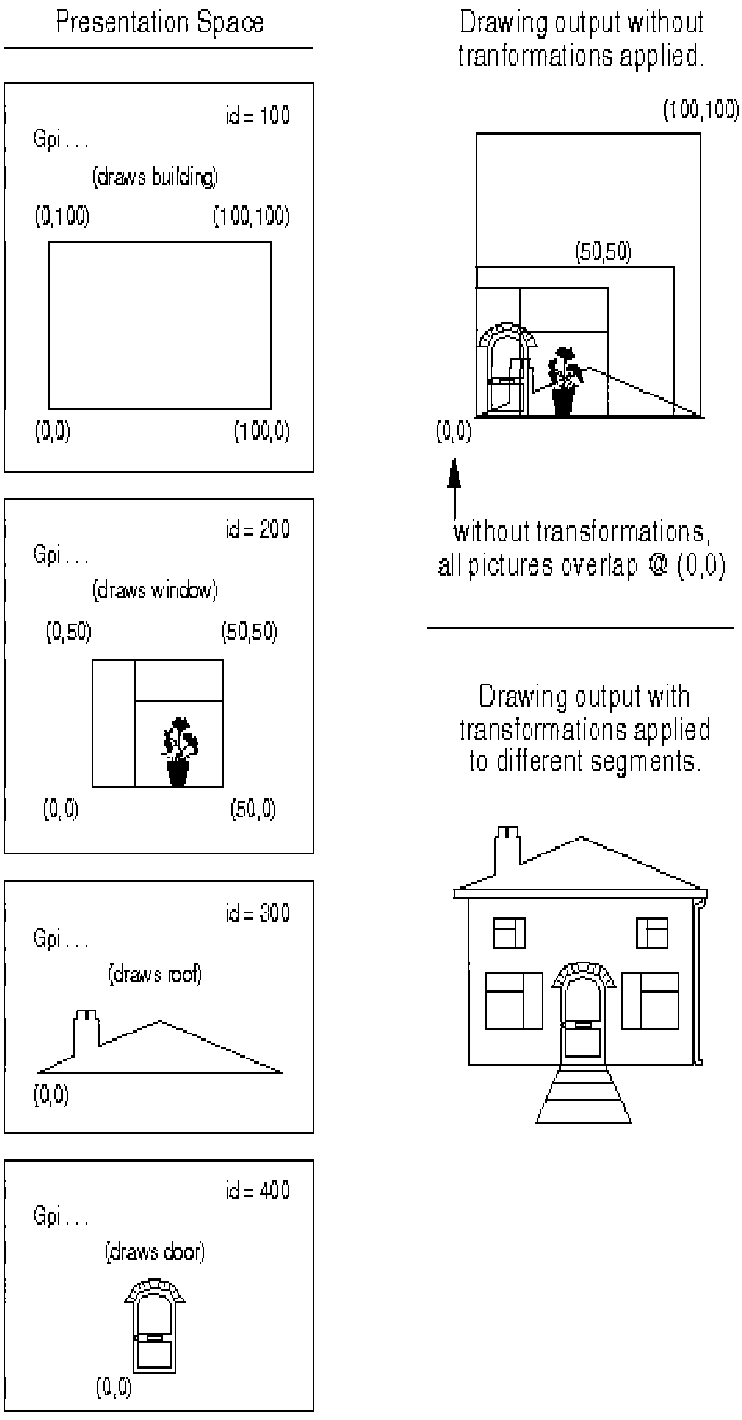
The device space is not affected by the PS_FORMAT option. Whether GPIF_LONG, or GPIF_SHORT, the device space is a rectangle with maximum coordinates of (32767, 32767) and minimum coordinates of (-32768, -32768). Note that not all of this rectangle is visible, since real devices are not that big. The device space relates to the physical device.

The transformation between the page and device space is handled automatically by the PM. The presentation page is mapped into a rectangle in the device space called the page viewport. This transformation is based on the parameters and options of GpiCreatePS.

Confusion can arise because the effects of the presentation page and the page viewport appear to be a clipping of the primitives in page and device space. While the action taken is identical, the term clipping is reserved for the activity deliberately designated by the application with clipping functions.

Applying Transformations Other Than the Identity Transformation

To better understand the workings of transformations other than the identity transformation, an example of the picture assembly process is illustrated in the following figure, and a detailed explanation follows.



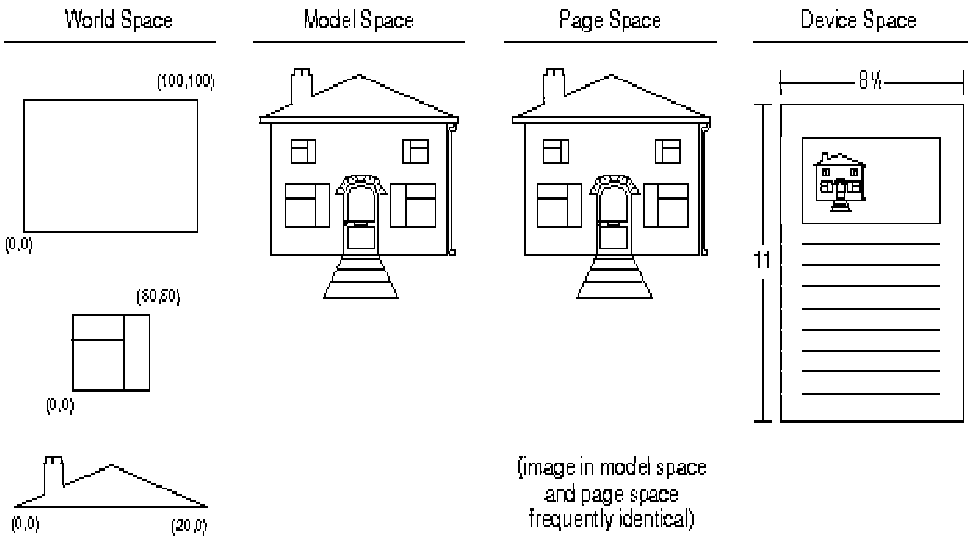
Picture Assembly Process

In the world space of the preceding figure four segments are drawn, each containing a different subpicture. The units of the subpictures can be different (for example, the building might be measured in feet, while the window might be measured in inches), because each subpicture is converted to the scale of the completed picture when it is transformed into the completed picture. All of the subpictures in the previous figure are defined at "real world" scales in their own (Cartesian) world coordinate space.

The difference between applying and not applying transformations can be seen in the model space in the previous figure. Without transformations, the subpictures would be drawn at exactly the coordinates given in the world space and thus all four subpictures would overlap. With transformations, the subpictures can be scaled and translated to the scale of the final picture. The window subpicture, Segment id=200 has been scaled, rotated, and translated.

The model space contains the model of what the application is trying to draw. There can be more than one model space for very complex drawings, but this is not a recommended style of programming.

The building also resides in the page space and is prepared in the device space for printing on an 8 and a half by 11 sheet of paper. This is shown in the following figure. Most often the page space to device space transformation (the device transformation) is the identity transformation.



Different Spaces

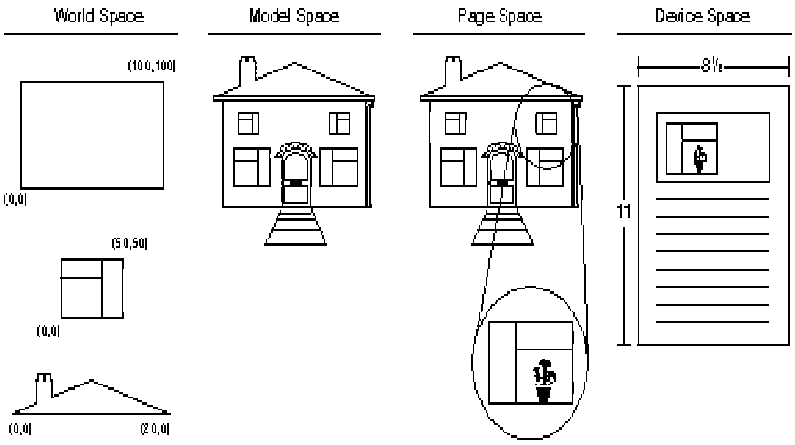
The transformation process in the previous figure is acting on segments. To draw the composite picture in model space, create a segment chain that plays all four segments associated with the building. The purpose of the world-to-model-space transformation is to transform the graphic segments into a composite picture.

Having built the composite picture, the next step is to map the composite into the page space. Usually, the model-to-page-space transformation is the identity transformation. The different coordinate spaces are shown in the previous figure.

The user might want an exploded diagram as well as the composite picture. This is illustrated in the following figure. To create the exploded diagram, the application must draw the segment chain again, with a slight translation down, and scaled up (enlarged) with a very large scaling factor.

Again, while it is possible to create several complex images in different model spaces that must be assembled in the page space, the most frequent use of the page space is to prepare a view of the model-from-model space with the first application of "real world" units such as inches or millimeters. As shown in the previous figures, both world and model space share unitless Cartesian coordinates.

The usual purpose of the page space is to show multiple views of the image residing in the model space, as shown in the following figure.



Different Spaces with an Exploded View

The enlarged image on the right side of the page space is drawn after the image on the left. It therefore has a higher priority and therefore would be drawn on top of the left side image. To obtain a multi-viewed page space image, the application defines a clipping region, and applies this region to the second (enlarged image) playback of the segment chain. The clipping region permits the enlarged image to appear only on the right side of the page space.

The same building also resides in this model space, but it resides in the page space, along with an exploded view of the uppermost right side window. The view is enlarged to such a magnitude that the details of the window are once again evident. If the images in world coordinate space are not able to be drawn in the amount of detail now supported by the PM, then drawing details, such as an exploded view, would reveal the barrenness of the image.

Subpictures can be drawn in world coordinate space without being drawn inside a graphics segment.

Combining Transformations Between a Coordinate Space Pair

There is more than one transformation between the world coordinate and model spaces, and between the model and page spaces. Depending on the desired output, the transformations can be combined in different ways.

Between the world coordinate and model space, there are five ways of combining the three transformations.

World Coordinate to Model Space Transformation Combinations

Transformation Function	Effect	Transformation Type Sequence		
GpiCallSegmentMatrix	ADD	M	I	S
	PREEMPT	I	M	S
	REPLACE		I	S
GpiDrawSegment (Call Segment)	Drawing outside of an instance drawing primitive but inside a segment.		M	S
No specific function	Drawing outside of segments altogether.			M

Abbreviations: I - Instance Transformation, M - Model Transformation, S - Segment Transformation

Between the model and page space pair are two transformations and they also can be combined in different ways.

Model to Page Space Transformation Combinations

Transformation Function	Effect	Transformation Type Sequence	
GpiSetPageViewport	Drawing inside of and outside of segments.	V	D
GpiSetDefaultViewMatrix	Drawing outside of segments.		D

Abbreviations: V - Viewing Transformation, D - Default Viewing Transformation

Transformation Mathematics

The transformation of a picture can be represented in general terms by two linear equations that define how the (x,y) coordinates of each point in the picture are changed.

```
The general form of these equations is:     where

x' = Ax + Cy + E                              (x,y) defines the original point
                                              (x',y') is the transformed point
y' = Bx + Dy + F                              A, B, C, D, E, and F are constants.
```

The transformations that result from these equations depend on the values of the constants, which in turn vary according to the type of transformation being applied.

The operating system handles transformations by using matrix mathematics. In the matrix mathematics,

- Translation is an addition operation
- Scaling, reflecting, rotation, and shear are multiplication operations.

If all the transformations were multiplication operations, different types of transformation could be applied with a single transformation operation. Therefore, to facilitate the combining of calls, translation in the IBM OS/2 is performed as a multiplication operation, rather than an addition operation.

This requires that the vector representing a point, [*x y*]
be extended by a third component, *w*: [*x y w*]
This enables all the transformations to be handled in a uniform manner.

This is called a *homogeneous coordinate system* . The value, *w* , is a multiplier, so that the point represented is: (*wx, wy*).

Note: The PM does not support a 3-dimensional presentation space. The 3-dimensional matrix is created to effect matrix multiplication.

In this notation, the point (*x, y*) is represented as [*x y 1*]. To be able to operate on such three-element vectors, and to combine translation with the multiplication operations, the 2-by-2 matrix has to be extended to a 3-by-3 matrix, with the third column being:

```
0
0
1
```

```
The linear equations:          become the matrix equations:

x' = Ax + Cy + E                                    A  B  0
                               [ x' y' 1 ] = [ x y 1 ] *  C  D  0
y' = Bx + Dy + F                                    E  F  1
```

In standard matrix notation, you have:

```
                              Msub11 Msub12 Msub13
[ x' y' 1 ] = [ x y 1 ] *     Msub21 Msub22 Msub23
                              Msub31 Msub32 Msub33
```

Model for Building the Transformation Matrix

The transformation matrix shown above can be set equal to an entity as complex as:

*M*I*S*V*D*

where * symbolizes matrix multiplication. As mentioned earlier, all the transformation types might not have a distinct value at all times. One, or all, could simply be the identity transformation.

These intermediate space and step-wise transformations are not actually how the operating system performs transformations. The reason for this relates to the way the PM stores transformation information. The transformation matrix data structure MATRIXLF, undergoes continuous modification during the drawing process, and it can be influenced by more than one transformation function from the application responding, for example, to what a user wants the drawing to look like.

For example, after creating the model of a building from drawing primitives, the user might want to see two identical buildings side-by-side. The application has already applied a series of transformations to the two segments in world coordinate space to create the model of the building. If the application, rather than the PM, had to perform the entire transformation over again from start to finish, it would have to keep track of much more detail. The PM enables applications to process a smaller portion of the MATRIXLF structure, in this example the "V" or "V and D" component, to enable the user to see two buildings side-by-side.

MATRIXLF, the Transformation Matrix

For all the transformation functions, the 3-by-3 transformation matrix is specified as a one-dimensional array in the following order:

(M11, M12, 0, M21, M22, 0, M31, M32, 1)

This one-dimensional array corresponds to the data structure, MATRIXLF. The elements in the data structure correspond to the matrix multiplication constants, as shown in the following table.

Transformation Type	Matrix Values
Scaling, Reflection	M11 and M22
Rotation	M11, M12, M21, M22
Translation	M31 and M32

All nine elements do not have to be specified for every transformation function. However, those specified are interpreted as the first *n* of the nine. If the third, sixth, and ninth elements are specified, they must be *0*, *0*, and *1* respectively.

MATRIXLF Structure

Of the nine fields in MATRIXLF, four are special 32-bit FIXED variables. The remaining five are 32-bit long integer variables.

The scaling, reflecting, and rotation constants, M11,M12,M21,M22, are the 32-bit, signed, FIXED variables, with an implied binary point between the second and third bytes. The translation constants, M31 and M32, and the three third-column variables, are 32-bit long integers.

A FIXED variable is a binary representation of a floating-point number. A FIXED variable has two parts:

- The high-order 16-bits, which contain a signed integer in the range -32768 through 32767.
- The low-order 16-bits, which contain the numerator of a fraction in the range 0 through 65535.

 The denominator for this fraction is 65536.

For example, to store the cosine of 60˚ (0.5) in a FIXED variable, an application would multiply 65536 by 0.5. The result, 32768, would be the value to assign to a field in the MATRIXLF structure.

To store a scaling factor of 3 in a FIXED variable, the application would multiply 65536 by 3. Again, the result, 196608, would be the value to assign to a field in the MATRIXLF structure.

MAKEFIXED Macro

The MAKEFIXED macro provides a quick and convenient method for setting the value of FIXED variables. This macro requires two arguments: the first is the integer part of the FIXED value, and the second is the fraction part of the FIXED value. In the following example, MAKEFIXED is used to assign the FIXED value equivalent of 1 1/8 to the matrix component M11.

```
matlf.fxM11 = MAKEFIXED(1, 8192)
```

The first argument, 1, is the integer part of the FIXED value. The second argument, 8192, is the result of multiplying 65536 by 1/8.

If it is necessary for an application to scale or rotate an object, the application can avoid most of these mathematics by using the helper functions, GpiRotate and GpiScale. GpiRotate accepts a rotation in degrees and converts this value into the appropriate fields in the MATRIXLF structure. Similarly, GpiScale accepts a scaling factor and fills in the MATRIXLF structure with the appropriate values.

About Transformation Operations

The available transformations are listed in the following table.

Transformations

Operation	Result
Scaling	Shrinks or enlarges the object
Reflection	Creates a mirror image of an object with respect to the x- or y- axis
Rotation	Rotates the object
Translation	Shifts the object with respect to the origin of the coordinate system
Shear	Rotates either all the vertical or all the horizontal lines in an object

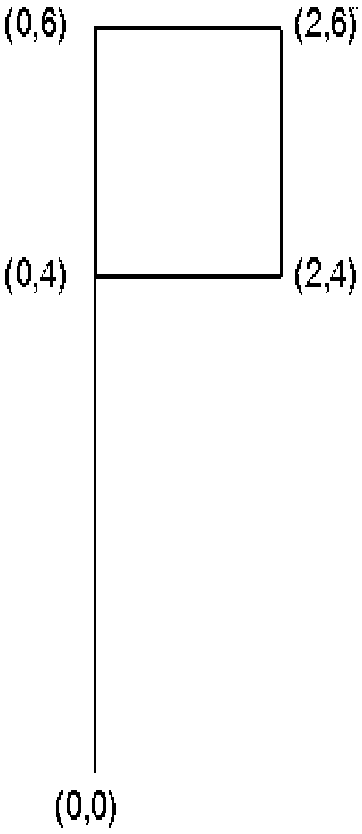
These basic operations can be combined.

The PM provides special functions to perform scaling, rotation, and translation and also enables applications to specify the transformation matrix directly. Applications can specify values for more than one type of transformation on a single transformation call.

Transformations are used to manipulate graphic objects as they are being moved from one coordinate space to another. These operations

are performed by functions called *transformation functions* . There are also functions that help perform the transformations called *helper functions* .

The scaling, reflection, rotation, translation, and shear transformations are best demonstrated by applying them to a picture. The following figure shows the image of a flag before any transformations have been applied. The flag is defined by five points. Their (x,y) coordinates are (0,0), (0,4), (0,6), (2,6), and (2,4).



Flag Before Transformation

In the next several sections, where the transformations are described in detail, the effect of the transformation on the flag is illustrated.

Scaling and Reflection Transformations

Applications can scale an object by using GpiScale or by modifying the MATRIXLF structure directly. A scaling transformation reduces or increases the size of a graphics object. A reflecting transformation creates a mirror image of an object with respect to the x- or y-axis.

A scaling factor of:

- greater than 1 causes an increase in size
- greater than 0 but less than 1 causes a reduction in size
- less than 0 causes a reflection about that axis. That is, a negative x scaling factor causes reflection in the x direction.

Note: If an application specifies a scaling factor of greater than 1, the graphics presentation space must be defined with the coordinate format GPIF_LONG. This is because 32-bit matrix elements are required to store these values in retained segment and metafile orders.

The equations to scale by factors Sx and Sy are obtained from the general equations (with M11 = Sx and M22 = Sy) and can be written:

$x' = xS_x$

$y' = yS_y$

A scaling transformation reduces or increases all the coordinates of an object by the scaling factor. Any object not aligned on the x- and y-axes is therefore moved nearer to the origin by a reduction in size, and away from the origin by an increase in size. For example, if an application applies a scaling factor of 0.5 to a simple box with its corners at (4,4), (10,4), (10,10), and (4,10), the four corners moves to (2,2), (5,2), (5,5), and (2,5).

To scale an object about a point without causing it to move, the following sequence of transformations is required:

1. Translate the scaling point of the object to the origin.
2. Scale the object at the origin.
3. Translate the scaling point of the object back to its original position.

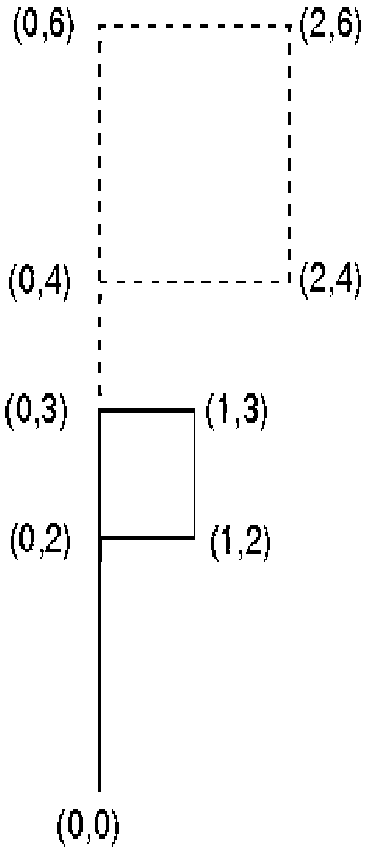
Scaling a Graphics Object

An application can scale the flag by 0.5, by applying:

$x' = 0.5x$

$y' = 0.5y$

The original five points of the flag are transformed:
(0,0) (0,0)
(0,4) (0,2)
(0,6) (0,3)
(2,4) (1,2)
(2,6) (1,3)
The following figure shows the effect of the scaling.



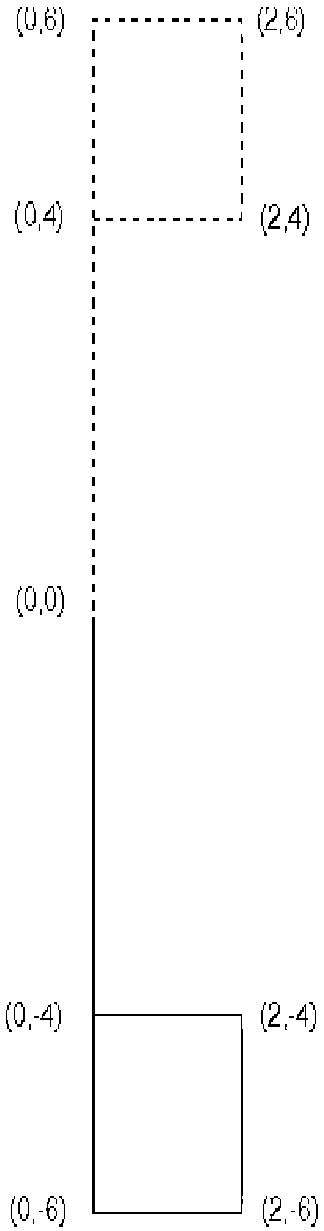
Scaling by 0.5

This scaling preserves the shape and orientation of the object, because the scaling factors in both directions are the same. However, scaling equations permit different scaling factors to be applied to x and y, which can cause distortion of the original shape of the object.

Reflecting a Graphics Object

A negative scaling factor causes a reflection of the object to be drawn. A scaling factor of -1, for example, causes a mirror image of the object to be drawn in the appropriate direction.

The following figure shows the flag reflected by applying a negative y scaling factor.



Reflection

MATRIXLF Structure for Scaling and Reflecting

When an application scales an object by using the scaling transformation, the matrix element M11 contains the horizontal scaling component (Sx), and the matrix element M22 contains the vertical scaling component (Sy).

```
                          Sx 0  0
[ x' y' 1 ] = [ x y 1 ] *  0  Sy 0
                          0  0  1
```

If the matrix element M11 contains a negative horizontal reflection component (-Sx), it causes reflection about the y-axis. If the matrix element M22 contains a negative vertical reflection component (-Sy) it causes reflection about the x-axis.

Rotation Transformations

The application can rotate an object either using GpiRotate or by modifying the MATRIXLF structure directly.

The operating system applies a transformation to all points in the source coordinate space. This means that unless an object is drawn about the origin of the source coordinate space, translation occurs when the object is rotated or scaled. GpiRotate enables an application to specify a point, relative to the origin, that is the center of rotation.

The equations for the rotation of an object about the origin (0,0) through an angle (theta), can be written:

x' = x cos (theta) - y sin (theta)

y' = x sin (theta) + y cos (theta)

A negative (theta) value rotates the object clockwise. For clockwise rotation, the rotation equations are:

x' = x cos (theta) + y sin (theta)

y' = -x sin (theta) + y cos (theta)

To rotate an object about some other point (p,q), the following sequence of transformations is required:

1. Translate the object by (-p,-q) to move the point of rotation to the origin.
2. Rotate the object around the origin.
3. Translate the object by (p,q) to move it back to its original position.

Rotation preserves the shape and size of the object.

Rotating a Graphics Object

An application can rotate the flag counterclockwise through 90°, by applying:

x' = x cos 90 - y sin 90

y' = x sin 90 + y cos 90

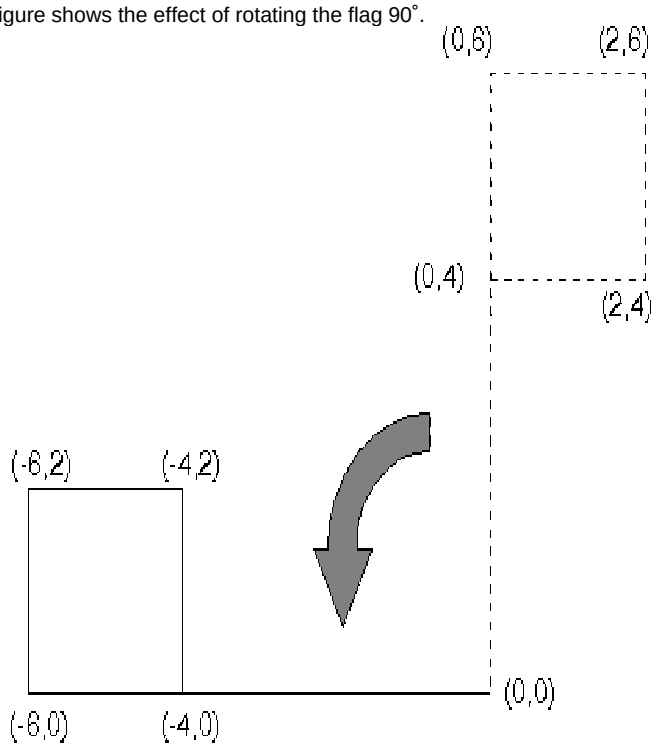
Because cos 90 = 0 and sin 90 = 1, these equations become:

x' = -y

y' = x

The original five points are transformed:
(0,0) (0,0)
(0,4) (-4,0)
(0,6) (-6,0)
(2,4) (-4,2)
(2,6) (-6,2)

The following figure shows the effect of rotating the flag 90˚.



Rotation Counterclockwise through 90˚

MATRIXLF Structure for Rotating

For counterclockwise rotation:

```
                                cos (theta)  sin (theta)  0
[ x' y' 1 ] = [ x y 1 ] *      -sin (theta)  cos (theta)  0
                                 0            0          1
```

For clockwise rotation:

```
                               cos (theta) -sin (theta)  0
[ x' y' 1 ] = [ x y 1 ] *      sin (theta)  cos (theta)  0
                                0            0          1
```

Translation Transformations

The application can translate an object either using GpiTranslate or by modifying the MATRIXLF structure directly.

To move a graphics object by an absolute number of coordinate units, a translation equation is applied. The equations for translation by Tx and Ty are obtained from the general equations (with E = Tx, and F = Ty) and can be written:

$x' = x + Tx$

$y' = y + Ty$

Translation preserves the shape, size, and orientation of the object. A negative Tx value causes movement to the left. A negative Ty value causes movement downward.

Translating a Graphics Object

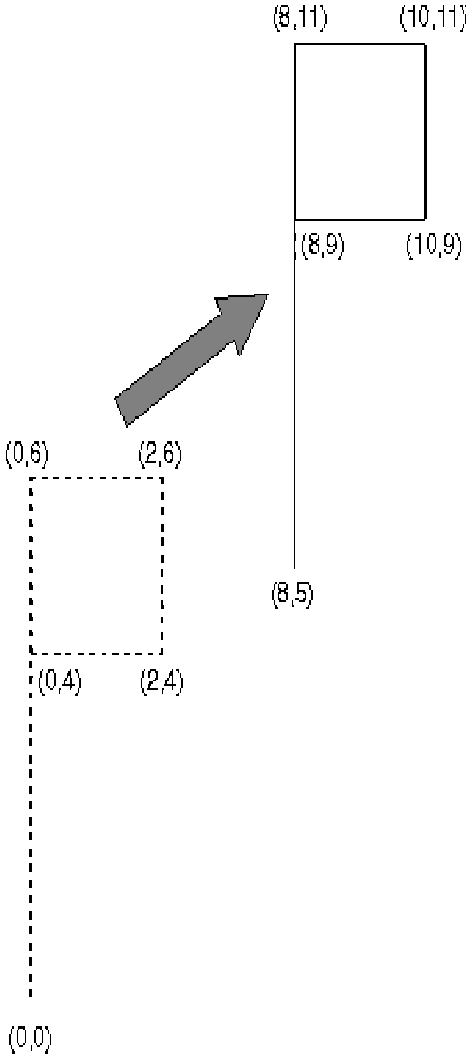
If Tx is equal to 8 and Ty is equal to 5, then:

$x' = x + 8$

$y' = y + 5$

The original five points are transformed:
(0,0) (8,5)
(0,4) (8,9)
(0,6) (8,11)
(2,4) (10,9)
(2,6) (10,11)

The following figure shows the effect of translating the flag by (8,5).



Translation by (8,5)

MATRIXLF Structure for Translating

When an application translates an object by using the translation transformation, the matrix element M31 contains the horizontal translation component, and the matrix element M32 contains the vertical translation component, as follows:

```
                                1  0  0
[ x' y' 1 ] = [ x y 1 ] *       0  1  0
                                Tx Ty 1
```

Shearing Transformations

There are two shear transformations: vertical and horizontal. The vertical shear transformation affects only the y-component of the coordinates of points in an object, and the horizontal shear transformation affects only the x-component.

If an application shears an object that contains two orthogonal vectors (two perpendicular lines), the vectors are no longer orthogonal.

A shearing transformation alters the shape of an object by translating its x-coordinates relative to its y-coordinates, or its y-coordinates relative to its x-coordinates. The amount by which the coordinates are translated is determined by the angle of the shear.

The equation for shearing an object to the left along the x axis by angle (theta) is:

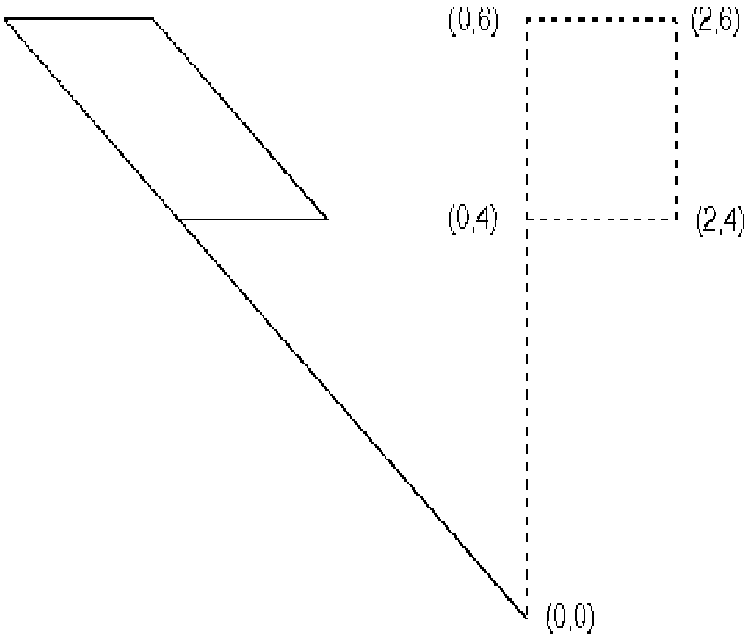
$x' = x - y$ tan (theta)

$y' = y$

To shear an object along the y-axis, the tangent of the angle of the shear is represented by constant B in the general equation.

Shearing a Graphics Object

The following figure shows the flag sheared to the left along the x-axis.



Shearing Along the X-Axis

MATRIXLF Structure for Shearing

For vertical shear transformation, the matrix element M11 contains the horizontal shear component, and the element M21 contains the vertical shear component, as follows:

```
                                1     -tan (theta)  0
[ x' y' 1 ] = [ x y 1 ] *       0           1       0
                                0           0       1
```

For horizontal shear transformation, the matrix element M21 contains the horizontal shear component, and the matrix element M22 contains the vertical shear component, as follows:

```
                                1           0  0
[ x' y' 1 ] = [ x y 1 ] *      -tan theta   1  0
                                0           0  1
```

About Transformation Functions

Transformation functions manipulate objects between coordinate spaces by applying transformations. Transformation functions require two coordinate spaces: a source coordinate space, and a target coordinate space. Which transformation function an application should use is determined by the two coordinate spaces and by the transformation effect desired.

The world-to-model space, model-to-page space, and page-to-device space transformations are all actually performed as a single operation. The different coordinate spaces are conceptual in nature, rather than explicitly defined entities. Describing them separately is meant to help explain the levels of activity during transformations.

All transformation functions share certain parameters, although the values and defaults of those parameters might not be identical.

Current Transformation

Every graphics object, whether in world, model, page, or device space, has a *current transformation* value, even if that value is simply the *identity transformation* .

The default current model transformation is the concatenation of any instance, segment, and model transformations from the root segment downward. The default current viewing transformation is the one most recently specified. The device transformation, which is set by the page viewport and the presentation page, should not be changed while drawing is in progress.

Accumulating Transformations

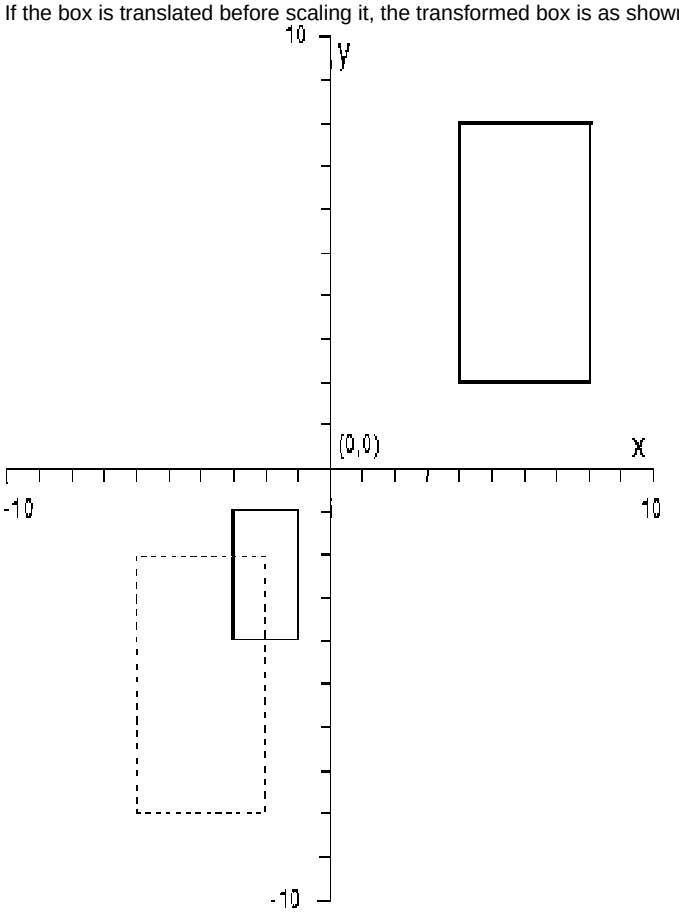
Each time one of the transformation functions is called by an application, the application can set the function's option parameter to control how the function combines the transformation with existing transformations and in what order the transformations are applied.

If the application uses this flag...	Then the operating system...
TRANSFORM_REPLACE	Replaces any existing transformations with the

	new transformation. The existing value of the matrix is discarded and replaced by straight substitution.
TRANSFORM_PREEMPT	Applies the new transformation before applying the existing transformation. The transformation matrix of the new transformation is pre-multiplied with the transformation matrix of the current transformation.
TRANSFORM_ADD	Applies the new transformation after applying the existing transformation. The transformation matrix of the new transformation is post-multiplied with the transformation matrix of the current transformation.

The order in which transformations are applied affects the appearance of the picture. For example, suppose that a box primitive has been defined, with its lower-left corner at (4,2) and its upper-right corner at (8,8), and that you want both to scale the box by 0.5 and to translate it by (-10,-10).

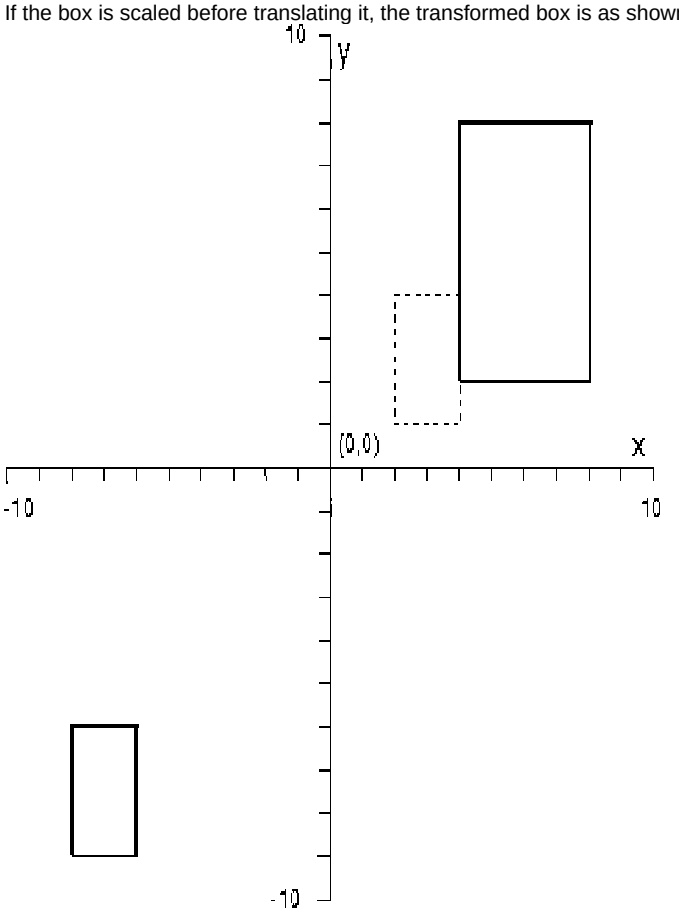
If the box is translated before scaling it, the transformed box is as shown in the following figure.



Translating before Scaling

The translated box has its lower-left corner at (-6,-8), and its upper-right corner at (-2,-2). Each of its coordinates is then scaled by 0.5, and the transformed box has its corners at (-3,-1), (-1,-1), (-3,-4), and (-1,-4).

If the box is scaled before translating it, the transformed box is as shown in the figure after the following figure.



Scaling before Translating

The scaled box has its lower-left corner at (2,1), and its upper-right corner at (4,4). The box is then translated by (-10,-10), and the transformed box has its corners at (-8,-6), (-6,-6), (-6,-9), and (-8,-9).

When an application is drawing a picture in which there are called segments, and in which transformations are applied to the root segments, the root-segment transformations should usually be applied to any segments they call. For example, if a segment that is translated to the left of the picture (by changing its segment transformation) calls a second segment, that leftward transformation should also be applied to the called segment. In this instance, the application would specify TRANSFORM_ADD in the call to GpiCallSegmentMatrix to add the instance transformation to the calling segments' segment transformation. Instance transformations are automatically reset on return to the calling segment.

Concatenating Transformations

When an application applies more than one transformation, it can concatenate the individual transformations to produce a final result. To concatenate transformations, multiply the individual transformation matrixes. The product of this multiplication is the *concatenated transformation* .

There are four ways the final matrix can be concatenated:

1. Hard code the matrix values into the application, then call the transformation functions.

2. Multiply the individual matrixes, then call the transformation functions.
3. Use the helper functions with the TRANSFORM_ADD option, then call the transformation functions.
4. Alternately apply transformation operations directly to the transformation matrix, then apply a transformation function.

Concatenating before calling provides better performance.

Hard Coding Values for a Concatenated Matrix

Pre-calculating the concatenated matrix values, then hard coding the values into the application, gives the fastest performance. However, this is rarely practical as the actual transformations to be performed are often variable.

If an application is, for example, performing interactive graphics, and one of the options offered to the user is a menu choice of rotate by 90˚, and enlarge by 4, then the rotational and scaling factors would not change, and the matrix values could be hard coded into the application.

Multiplying Matrix Values

Multiplying transformation matrix values directly offers the second fastest performance, yet can still respond to a variety of desired transformations. As many transformation matrixes as needed can be multiplied together. If the application is concatenating the matrixes itself, it is responsible for preventing the accumulated transformation side effects.

For example, to rotate an object counterclockwise about the point (p,q) using a single transformation call requires three transformations to be concatenated. When the application is specifying each transformation, step-by-step, the sequence of actions would be:

1. Translate the object by (-p,-q) to move the point of rotation to the origin.
2. Rotate the object about the origin.
3. Translate the object by (p,q) to move it back to its original position.

The individual matrixes are:

```
 1   0  0      cos (theta)  sin (theta)  0     1  0  0
 0   1  0     -sin (theta)  cos (theta)  0     0  1  0
-Tx -Ty 1       0            0           1     Tx Ty 1
```

The matrix for the concatenated transformation is produced incrementally. That is, two adjacent matrixes are multiplied to produce a single matrix, which is then multiplied with the third matrix. You can begin by multiplying either the first two matrixes or the second two matrixes. If you start by multiplying matrixes 2 and 3 together, the resulting matrix is:

```
            cos (theta)  sin (theta)  0
           -sin (theta)  cos (theta)  0
             Tx           Ty          1
```

This matrix is multiplied with the first matrix to produce the matrix for rotating an object at the point (p,q):

```
 cos (theta)  sin (theta)  0
-sin (theta)  cos (theta)  0
  a            b           1
```

where:

a = (-*Tx*cos (theta) + *Ty*sin (theta) + *Tx*)

and

b = (-*Tx*sin (theta) - *Ty*cos (theta) + *Ty*)

If an application were performing the concatenation for a scaling operation, again it would have to specify the transformation step-by-step. The sequence of actions would be:

1. Translate the object's scaling point to the origin.
2. Scale the object at the origin.
3. Translate the object back to its original position.

Here are the three matrixes required to obtain this effect:

```
 1   0  0      Sx  0   0     1  0  0
 0   1  0      0   Sy  0     0  1  0
-Tx -Ty 1      0   0   1     Tx Ty 1
```

The matrix of the concatenated transformation is:

```
      Sx          0       0
      1           Sy      0
(-TxSx+Tx)  (-TySy+Ty)  1
```

Transformation Helper Functions

Three helper functions, are provided to perform the matrix math required to concatenate transformations:

- GpiTranslate
- GpiRotate
- GpiScale

Any of these three functions can be used with the TRANSFORM_ADD option to concatenate the new matrix with an existing matrix. This method builds up the matrix in application storage in a sequence of steps before using a single transform function.

While this is slower than hard coding the matrix, it is faster than alternating between applying transformation operations directly to the matrix then applying a transformation function.

The helper functions merely calculate the appropriate matrix. The transformation is not applied until the array containing the matrix values is passed to the appropriate transformation function.

Applications use GpiTranslate to change the position of an object. The application specifies the coordinates of the point to which to move the object and the name of the transform matrix to use as input to GpiTranslate. The transformation matrix must be in the form of a one-dimensional array. The application also can specify whether this transformation is to replace the value for a previous transformation, or whether it is to be added to it.

Applications use GpiRotate to rotate an object. The application specifies the angle of rotation, the coordinates of the point around which the object is to rotate, and the transformation matrix. The transformation matrix must be in the form of a one-dimensional array. The application also can specify whether this transformation is to replace the value for a previous transformation, or whether it is to be added to it.

To scale an object at a point without also moving the object, applications use GpiScale. When using GpiScale, the application specifies the scaling factor, the coordinates of the center point, and the transformation matrix. The transformation matrix must be in the form of a one-dimensional array. The application also can specify whether this transformation is to replace the value for a previous transformation, or whether it is to be added to it.

An application could alternate between applying transformation operations directly to the transformation matrix, then applying a transformation function with the TRANSFORM_ADD option set. This would build up the matrix in the presentation space. This method has the slowest performance of the concatenation methods.

Round-Off Error

Whenever an application uses transformations, it should handle any round-off error that occurs after multiple scaling, rotation, shear, or reflection transformations. The rounding error increases because a transformation is incrementally updated by the amount of the error with, for example, the TRANSFORM_ADD option. For the rotation to remain accurate, the application should recalculate the transformation, rather than accumulate many small changes.

For example, if an application uses a rotation transformation to rotate the hands of a clock, the accuracy of the clock diminishes due to rounding off after the transformation. The rounding error should be periodically removed by using the TRANSFORM_REPLACE option at known points, for example, every 90˚ or every complete revolution of the clock.

World-Space to Model-Space Transformations

The model transformation drawing attribute operates between world and model space. This attribute can be updated by one of the three following transformation functions:

- GpiSetModelTransformMatrix
- GpiSetSegmentTransformMatrix
- GpiCallSegmentMatrix

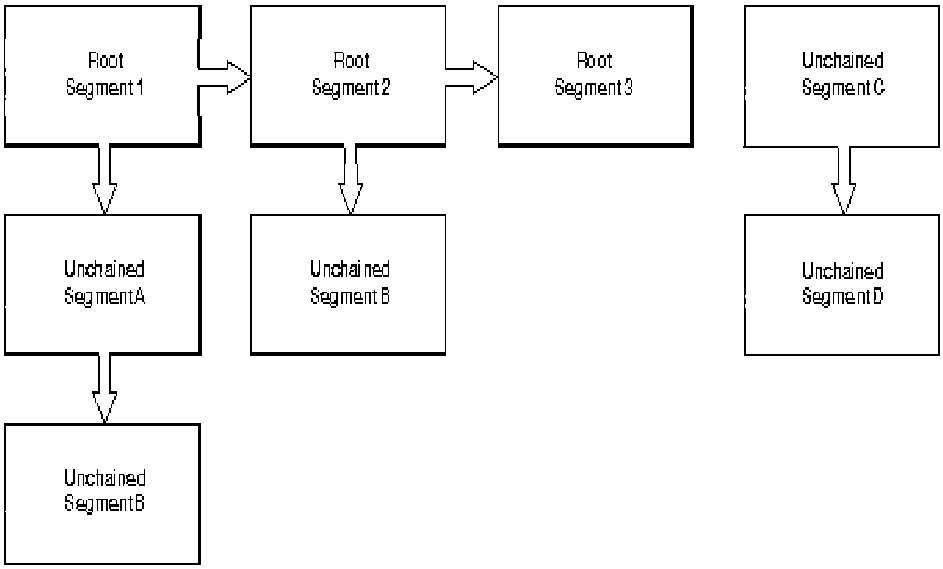
If the model transformation drawing attribute has never been updated, the attribute defaults to the identity transformation. If the drawing attribute has been updated, the *existing transformation* is the concatenation of any instance, segment, or model transformations from the root segment downwards. Each time a new segment is opened (using GpiOpenSegment), the model transformation drawing attribute is reset to its default value.

The three transformations that operate between world space and model space are:

- Model transformations
- Segment transformations
- Instance transformations

The transformations that occur between world-coordinate and model spaces depend on the drawing mode and the possible segment type of the drawing primitive.

The drawing mode can be either nonretain or retain. Nonretain mode is also called draw mode, as the graphics are immediately displayed. In retain mode, the graphics orders are stored in chained and unchained segments. A series of segments are shown in the following figure.



Segments

A model transformation effects objects in any of these segments. A segment transformation affects the six chained segments, on the left and must be issued before Root Segment 1 is accessed. An instance transformation can be applied only to Segment B or Segment D and must be issued from Segment A or Segment 2 or Segment C. The instance transformation is reset on return to the calling segment.

Model Transformations

Model transformations also can be applied to objects both inside and outside of segments. Model transformations also can be applied to objects created under retained or nonretained drawing mode. This means model transformations affect the output from:

- GpiDrawChain
- GpiDrawFrom
- GpiDrawSegment
- Any GPI functions that occur outside of a retained-drawing segment

Applications can change the model transformation any number of times in a single segment. Changing the model transformation changes the transformation applied to any drawing primitives used subsequently.

An application can determine the values for the current model transformation by calling GpiQueryModelTransformMatrix, model transformation's scaling, rotation, and translation values in a one-dimensional array representing elements in the MATRIXLF structure.

GpiSetModelTransformMatrix specifies the model transformation's scaling, rotation, and translation values applied to subsequent primitives in the segment. As this function does not require the name a segment, it also can be used to apply transformations to primitives outside of segments.

For example, in the building example earlier in the chapter, if the window is stored in a retained segment, an application could draw several identical windows on the house by setting a new translation component in the model transformation before calling GpiBox.

Segment Transformations

The segment transformation provides a way of defining the model transformation drawing attribute at the start of a retained segment. The attribute can subsequently be updated by the model transform orders within the segment. A segment transformation can be applied only to a

retained segment.

Segment transformations alter retained-drawing output. Unlike a model transformation, which can be set and reset within a segment bracket, a segment transformation must be set outside of a segment bracket.

An application can determine the values for the current segment transformation by calling GpiQuerySegmentTransformMatrix, which returns the model transformation's scaling, rotation, and translation values in a one-dimensional array representing elements in the MATRIXLF structure.

To apply a segment transformation, applications use GpiSetSegmentTransformMatrix. GpiSetSegmentTransformMatrix can be used to apply any sort of transformation to a segment. For example, it can be used to translate a dynamic segment from one screen position to another, when movement is requested by the user. The application could perform the following steps, for example, on receipt of a WM_MOUSEMOVE message:

1. Use the new mouse position to calculate the required displacement from the current position.
2. Call GpiRemoveDynamics to remove the dynamic segment from its current position.
3. Call GpiSetSegmentTransformMatrix to translate the segment coordinates by the required amount.
4. Call GpiDrawDynamics to redraw the segment in its new position.

Instance Transformations

Instance transformations provide a way of defining the model transformation drawing attribute for the duration of a called segment.

Instance transformations alter the retained-drawing output from special segments referred to as *called* segments. A called segment usually contains a subpicture duplicated several times in other subpictures. The instance transformation positions, sizes, and rotates the subpicture each time the segment is duplicated, because the transformation is set each time the segment is called. An instance transformation applies only to the called segment, and is reset on return to the calling segment. There is no query function associated with the transformation as it must be explicitly set for each called segment.

To apply an instance transformation, applications call GpiCallSegmentMatrix from the calling segment. GpiCallSegmentMatrix calls the segment and also applies the model transformation's scaling, rotation, and translation values in a one-dimensional array representing elements in the MATRIXLF structure and passes a segment identifier.

World Space-to-Model Space Transformation Summary

The following table summarizes the choices available during world-coordinate space to model space transformations.

World to Model Space Transformation Summary

Transformation Type	Transformation Function	Applies to...
Model transformations	GpiSetModelTransformMatrix	Primitives both inside and outside a segment
Segment transformations	GpiSetSegmentTransformMatrix	A retained segment, issued at the first element of the segment
Instance transformations	GpiCallSegmentMatrix	Only the called segment

Note: The effect of GpiSetModelTransformMatrix and GpiCallSegmentMatrix on the model transformation drawing attribute can be duplicated by an equivalent drawing order in a drawn retained segment.

Model Space-to-Page Space Transformations

There are two transformations that operate between the model space and the page space:

- Viewing transformations
- Default viewing transformations

All three model transformation types and the viewing transformations can be considered a part of the picture. The default viewing transformation is part of the environment, and must not be used for picture construction.

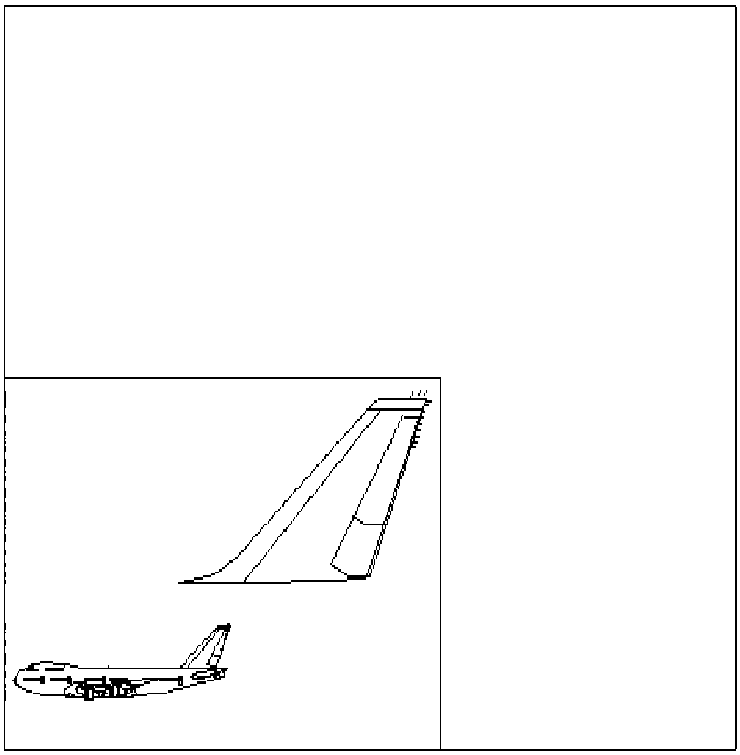
Viewing transformations only apply in retained or nonretained segments. The viewing transformation attribute is set to the presentation space viewing transformation matrix value at the start of each drawn root segment and remains constant for that segment. If GpiSetViewingTransformMatrix is called, the new value is not used until the next segment is opened. The matrix drawing attribute is reset to identity at the end of the segment. Each change in a viewing transformation is equivalent to defining a new model space.

The default viewing transformation is a presentation space attribute and should not normally be modified in the middle of a picture. This attribute can be updated by GpiSetDefaultViewMatrix.

A picture is normally constructed in the presentation page with an identity default viewing transformation. The default viewing transformation can then be used to scale and scroll the entire picture in the presentation page.

Viewing Transformations

More than one copy of an entire model space can be drawn in page coordinate space, and each copy can be transformed as required. Parts of the model space also can be transformed to the presentation page. For example, an enlarged view of the tail of an aircraft can be shown with a reduced view of the complete aircraft in one corner. This is shown in the following figure.



Presentation-Page Space

The entire model space (the aircraft) and a part of the model space (the tail of the aircraft) are drawn to a single page coordinate space. In each instance, scaling and translation transformations have been applied.

Alternatively, the displayed picture can be made up of several subpictures with no common graphical elements. For example, the aircraft can be drawn in one part of the display, and a map of an airport in another part of the display. In this instance, the final picture would be derived from different model spaces.

Whether multiple views are derived from a single model space or from different model spaces, there are two items to address for each instance of a model space incorporated into the presentation page:

1. The part of the model space to be displayed must be identified by defining a *viewing window* for the model space.
2. To position and size the contents of the model space in page coordinate space, a viewing transformation must be specified.

To get views of one or more model spaces on the screen simultaneously, each model space is drawn the required number of times. Before each drawing request, the viewing window is defined and a viewing transformation is specified.

Defining the Viewing Window

The viewing window is a conceptual boundary around a part of the model space. To produce the picture in the previous figure, the aircraft is drawn twice. The first time, the viewing window is on only the tail of the aircraft, and the second time, the viewing window is on the entire aircraft. Only those parts of the model space within the viewing window are visible in the picture assembled in presentation-page space. GpiSetViewingLimits is used to define the viewing window.

Graphics Field

Applications also can specify a type of viewing window for the presentation page smaller than the page. This window is known as the *graphics field*. To define a graphics field, use GpiSetGraphicsField. By default, no graphics field is specified. If a graphics field is defined, the picture assembled within it is the picture that is visible on the output device.

An application can determine the current values for the viewing transformation by calling GpiQueryViewingTransformMatrix, which returns the transformation values in a one-dimensional array representing elements in the MATRIXLF structure. The application can set the values by calling GpiSetViewingTransformMatrix, and passing the transformation values in a one-dimensional array representing elements in MATRIXLF structure.

Default Viewing Transformations

The default viewing window is the same size as the model space. Therefore, to display one or more entire model spaces, draw the picture the required number of times and let the viewing window default each time.

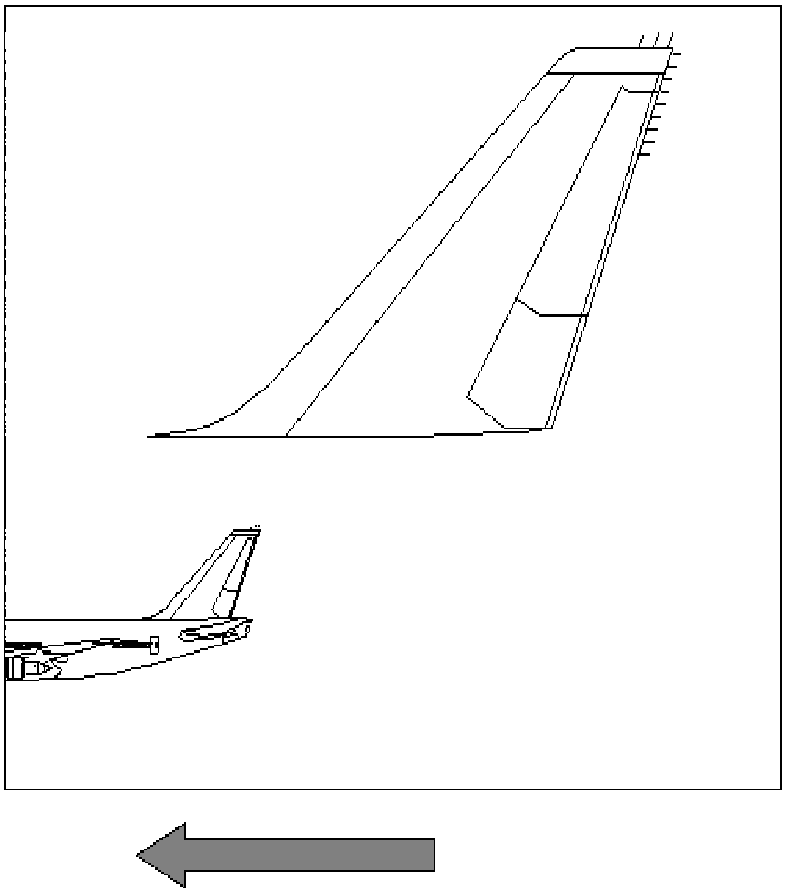
Default viewing transformations scroll or zoom pictures in a window on a display screen. An application can determine the current values for the default viewing transformation by calling GpiQueryDefaultViewMatrix, which returns the default-viewing-transformation values in a one-dimensional array representing elements in MATRIXLF structure. The application can set these values by calling GpiSetDefaultViewMatrix and passing the transformation values in a one-dimensional array representing elements in MATRIXLF structure.

A default viewing transformation is applied when the screen contents are zoomed or scrolled by user interaction. A picture is zoomed when the user wants to increase or decrease the size of an area of interest. A picture is usually scrolled when there is more in the presentation page than can be displayed in a single page of output. Anything lying off the screen, but within the range of the presentation page, can be scrolled into view. The default viewing transformation applies to the entire page coordinate space, and can be added to, or can replace, the current default viewing transformation. The PM applies it after any viewing transformations.

When a presentation page is created, the default viewing transformation is set to identity. For example, if the presentation-page contents are scrolled:

- Erase the screen contents.
- Call GpiSetDefaultViewMatrix to translate the presentation-page picture by the required amount.
- Draw the picture again.

The following figure shows the airplane presentation-page contents scrolled to the left.



Scrolling the Presentation Page

Every presentation-page coordinate is translated to the left by the same amount.

Zooming is implemented in the same way, except that the default viewing transformation is used to scale the picture up or down as required.

If you want to display only one view of a single picture, and if you do not want scrolling and zooming capabilities, you can let the viewing and default viewing transformations default. When both transformations are permitted to default, page coordinate space is effectively the same as model space.

Page-Space to Device-Space Transformations

There is only one transformation between the page coordinate space and the device space, the *device transformation* . The device transformation enables applications to work in any presentation-page units regardless of the target device. Unlike the transformations previously described, the device transformation only scales and translates objects, and it is defined by two rectangles.

The first rectangle is called the *presentation page* . Its location is the bottom left origin and its dimensions are fixed. The second is called the *page viewport* . Its location is the bottom left origin and its dimensions can be varied.

The device transformation, which maps the picture in presentation-page space to device space, happens automatically. The device transformation is established when the presentation space is created, and ensures that graphics are displayed in the correct size and, where possible, that their aspect ratio is preserved.

To modify the device transformation, applications use GpiSetPageViewport. Input for this function is the device coordinates of the lower-left and upper-right corners of the page viewport. Applications should modify the default device transformation only when it is necessary to use nonstandard page units.

Presentation Pages

A presentation page is a rectangle in a page space. Its lower-left corner is always positioned at the origin of the page space.

An application can determine the dimensions of the presentation page by calling GpiQueryPS. It returns a pointer to a SIZEL structure that contains the page dimensions. If an application specifies arbitrary page units when creating a presentation space with GpiCreatePS, the page viewport is constructed such that the origin of the page rectangle maps to the origin of the default device rectangle and either the right or top edge maps to the corresponding edge. Thus, the aspect ratio of the graphic is preserved.

If either the height or width of the presentation page is set to 0 (using GpiCreatePS), the application must set GPIA_ASSOC to set the default presentation page size to the default device rectangle size.

Page Viewports

A page viewport is a rectangle in device space, whose origin and size can be varied. The operating system uses the presentation-page rectangle and the page-viewport rectangle to define the device transformation.

An application can determine the current dimensions of the page viewport by calling GpiQueryPageViewport, which returns a pointer to a RECTL structure that contains the coordinates of the viewport. The application can set the location and dimensions of the page viewport by calling GpiSetPageViewport and passing it a pointer to a RECTL structure that contains the new values.

The ratio of the page width to the page-viewport width defines a horizontal scaling factor. The ratio of the page height to the viewport height defines a vertical scaling factor. Applications can use DevQueryCaps to obtain the horizontal and vertical resolution of a device in pels per meter. The dimensions of a pel can vary from one device to another, but they usually fall in the range of 0.25 to 0.50 mm.

The page viewport can be shifted in the device space by a translation transformation.

Mapping the Presentation Page to the Device

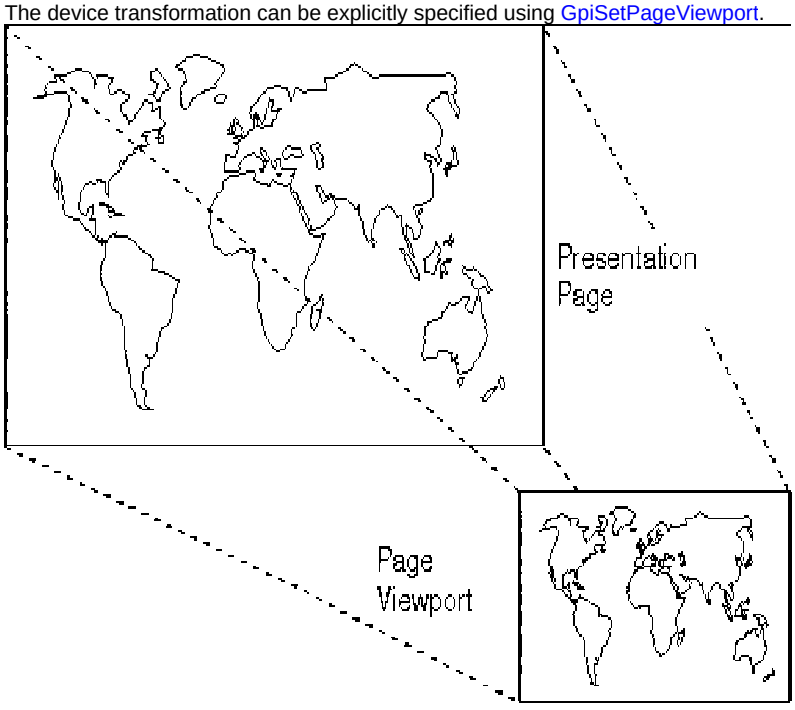
When an application associates a presentation space with a device context, a default device transformation is set. A page viewport is defined according to the rules in the following table:

Presentation-page specification	Page viewport size	Usage.
Pels	The same size as the presentation page.	The lower-left corner of the presentation page maps to the lower-left corner of the device space. For example, if an application defines a presentation page of 300 coordinates (x-axis)-by-200 coordinates (y-axis), then the picture is transformed to a screen area of the same size.

Metric units	The coordinates that produce the correct matrix for the physical spacing of the pels.	The lower-left corner of the presentation page maps to the lower-left corner of the device space.
Arbitrary units	The default size for the device is used. For a plotter or printer, this is the maximum accessible area of the paper, and for a screen, it is the maximized window size.	The page viewport is constructed such that the presentation-page coordinates give equal x- and y-spacing. The lower-left corner of the presentation page maps to the lower-left corner of the device space, and either the right or the top edges map, such that the picture is contained within the device rectangle and its aspect ratio is preserved.

The following figure shows mapping from the presentation page to the device.

The device transformation can be explicitly specified using GpiSetPageViewport.



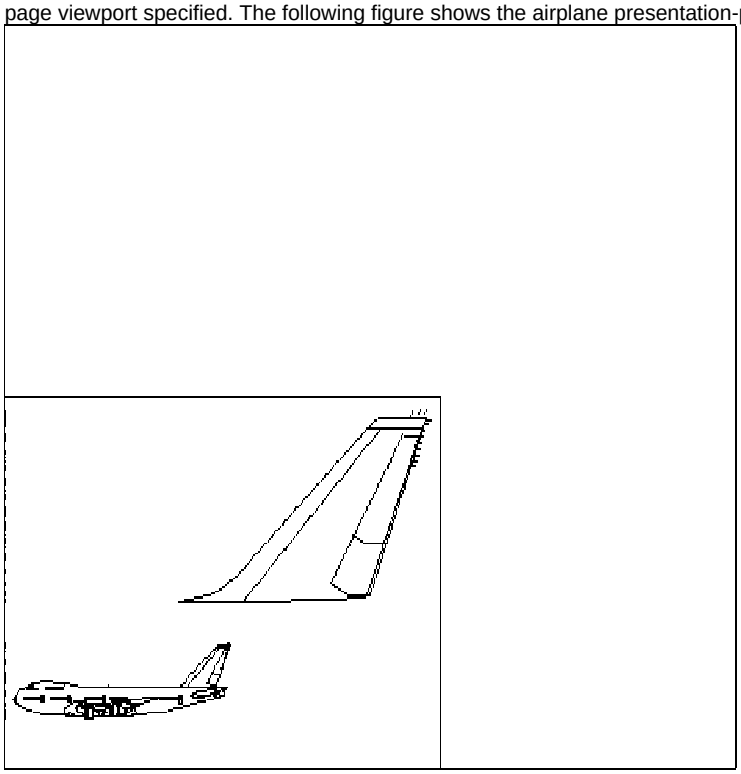
Mapping a Picture from the Presentation Page to the Device

In this example, a map of the world has been drawn in a presentation page, defined in arbitrary units, that is much larger than the device space. The device transformation scales the picture to fit the maximized window size and preserves its aspect ratio.

Coding the Device Transformation

The PM automatically transforms the presentation-page contents to the area of the device space within the page viewport. The drawing is not clipped to the page viewport because this is a scaling transformation only. The entire picture is displayed, regardless of the size of the

page viewport specified. The following figure shows the airplane presentation-page contents scaled to fit the page viewport.



Device Space

A page viewport smaller than the presentation page has been defined. The picture assembled in the presentation page is therefore scaled to fit the page viewport.

After transformation to device space, graphics coordinates must be in the range -32768 through +32767, even if the presentation page is defined in GPIF_LONG format. An attempt to address a coordinate outside this range results in a coordinate-overflow error. To determine if a graphics object will give an error, applications can do the following:

- If the application is not rotating or shearing a graphics object, it calls GpiConvert to convert the device-space limits to world-coordinate-space limits, then uses these limits when creating the graphics object.
- If the application is rotating or shearing a graphics object, it uses GpiConvert to convert the device-space limits back to model space, and ensures that the picture boundary is inside these limits. Note that this method only applies if all rotational and shearing is performed using 1 of the model transformation types.

Remember that world-coordinate space has its own own limits:

- -32768 through +32767 for a GPIF_SHORT-format presentation page
- -134217728 through +134217727 for a GPIF_LONG-format presentation page.

Although applications can specify a page viewport of any size, the presentation page can be mapped only to an area equal to, or less than, the available device space. If an application specifies a viewport larger than the available device space, part of the presentation page contents are displayed outside the visible device output area. To find out the dimensions of the page viewport for the currently associated device, use GpiQueryPageViewport. Applications can store the dimensions of the current page viewport before changing them, and restore them later.

Device-Transformation Matrix

To directly manipulate the device-transformation matrix it is necessary for applications to know the following values:

X1 the x-coordinate of the lower-left corner of the page viewport

Y1 the y-coordinate of the lower-left corner of the page viewport

X2 the x-coordinate of the upper-right corner of the page viewport

Y2 the y-coordinate of the upper-right corner of the page viewport

XPS the presentation-space width -1

YPS the presentation-space height -1.

The device-transformation matrix is defined as:

```
The device-transformation matrix:    where:

  Msub11  0     0                   a = (Xsub2-Xsub1)/(Xps+1)
   0     Msub22 0
  Msub31 Msub32 1                   b = (Ysub2-Ysub1)/(Yps+1)

                                    c = Xsub1+(a-1)/2

                                    d = Ysub1+(b-1)/2
```

Windowing-System Transformation

There is one transformation, the *windowing-system transformation* , performed automatically by the PM. The windowing-system transformation, which is a translation transformation only, maps the coordinates of the picture in device space to the coordinates of the screen window or printer page. This happens when a picture is first drawn and whenever the display window in which the picture has been drawn is moved.

Transforming Bit-Map Data

In general, graphics defined in device coordinates (bit maps and image primitives) cannot be transformed. For example, the size of an image primitive is specified in device coordinates, and cannot be altered. The size, therefore, remains unaltered down the viewing pipeline. The position of an image primitive, however, is specified in world coordinates. The image is therefore subject to translation transformations. Note, however, that GpiWCBitBlt permits the target rectangle to be specified in world coordinates, which are transformed.

Because the position of the image primitive is specified in world coordinates and its width is specified in device coordinates, positioning two images together on the screen causes special difficulties. The second image cannot be positioned without knowing the width, in world coordinates, of the first image. To get the width of the first image:

1. Identify two coordinate positions, one on the image's left edge, and one on its right edge. For example, the two positions could be (10,80) and (150,80). These positions are in device coordinates.

2. Convert these device coordinates to world coordinates using GpiConvert. GpiConvert converts an array of (x,y) coordinates that apply in one coordinate space to their corresponding values in another coordinate space.

3. Subtract the lower x-coordinate from the higher x-coordinate. In the above example, the width of the image is the difference between the world-coordinate equivalents of 150 and 10.

When you have the width of the first image in world coordinates, you can calculate the start position of the second image.

Paths, although defined in world coordinates, are device-dependent and are bound in device coordinates when they are defined. Subsequent transformations (other than the windowing-system transformation) have no effect on paths. However, if a path is used to create

a wide line, the width of the line is scaled as required.

Using Coordinate Spaces and Transformations

This section explains how to:

- Set an application's drawing units to convenient units
- Translate, rotate, and scale a picture
- Shear a picture

Setting Drawing Units

Applications can use GpiCreatePS to set the device transformation so that it uses page units that might be more convenient than pels; for example, centimeters. If the output device is a screen, the application first opens a device context by calling WinOpenWindowDC. If the output device is a printer or plotter, the application opens a printer or plotter device context by calling DevOpenDC. The application then creates a presentation space by calling GpiCreatePS, specifying low-metric page units and associating the device context with the presentation space. The following figure is an example of how to set drawing units.

```
HWND hwnd;                    /* Client-window handle         */
HAB hab;                      /* Anchor-block handle          */
HPS hps;                      /* Presentation-space handle    */
HDC hdc;                      /* Device-context handle        */
SIZEL sizlPage;               /* Presentation-page rectangle  */

hdc = WinOpenWindowDC(hwnd);
sizlPage.cx = 0;
sizlPage.cy = 0;
hps = GpiCreatePS(hab,        /* Anchor-block handle          */
      hdc,                    /* Device-context handle        */
      &sizlPage,              /* Address of SIZEL structure   */
      PU_LOMETRIC             /* Centimeters as page units    */
      | GPIA_ASSOC);          /* Associates window DC with PS */
```

Translating, Rotating, and Scaling a Picture

GpiTranslate, GpiRotate, and GpiScale provide a convenient method for transforming objects in a picture. The following figure shows how to use these functions to translate, rotate, and scale a triangle.

```
MATRIXLF matlfTransform;
POINTL ptlStart, ptlTrans, ptlRotate, ptlScale;
FIXED fxAngle, afxScale[2];
POINTL aptlTriangle[] = { 575, 300, 575, 500, 500, 300 };

ptlStart.x = 500;                    /* Starting point x direction   */
ptlStart.y = 300;                    /* Starting point y direction   */
GpiMove(hps, &ptlStart);
```

```
GpiPolyLine(hps, sizeof(aptlTriangle) / sizeof(POINTL), aptlTriangle);

ptlTrans.x = 75;                        /* x coordinate for translation */
ptlTrans.y = 75;                        /* y coordinate for translation */

GpiTranslate(hps,                       /* Presentation-space handle    */
    &matlfTransform,                    /* Address of matrix            */
    TRANSFORM_REPLACE,                  /* Replace old matrix with new  */
    &ptlTrans);                         /* Coordinates for translation  */

GpiSetModelTransformMatrix(hps,         /* Presentation-space handle    */
    9,                                  /* Number of points in matrix   */
    &matlfTransform,                    /* Address of matrix            */
    TRANSFORM_REPLACE);                 /* Replace old matrix with new  */

GpiMove(hps, &ptlStart);                /* Move to starting point       */
GpiPolyLine(hps, sizeof(aptlTriangle) / sizeof(POINTL), aptlTriangle);

fxAngle = MAKEFIXED(-45, 0);            /* Rotate 45 degrees clockwise  */
ptlRotate.x = 550;                      /* x coordinate rotation origin */
ptlRotate.y = 350;                      /* y coordinate rotation origin */

GpiRotate(hps,                          /* Presentation-space handle    */
    &matlfTransform,                    /* Address of matrix            */
    TRANSFORM_REPLACE,                  /* Replace old matrix with new  */
    fxAngle,                            /* Rotation angle               */
    &ptlRotate);                        /* Origin of rotation           */

GpiSetModelTransformMatrix(hps, 9, &matlfTransform, TRANSFORM_REPLACE);

GpiMove(hps, &ptlStart);                /* Move to starting point       */
GpiPolyLine(hps, sizeof(aptlTriangle) / sizeof(POINTL), aptlTriangle);

ptlScale.x = 550;                       /* x coordinate scale origin    */
ptlScale.y = 350;                       /* y coordinate scale origin    */
afxScale[0] = MAKEFIXED(2, 0);          /* Scaling factor on x axis     */
afxScale[1] = MAKEFIXED(2, 0);          /* Scaling factor on y axis     */

GpiScale(hps,                           /* Presentation-space handle    */
    &matlfTransform,                    /* Address of matrix            */
    TRANSFORM_REPLACE,                  /* Replace old matrix with new  */
    &afxScale[0],                       /* Scaling factor               */
    &ptlScale);                         /* Origin of scaling operation  */

GpiSetModelTransformMatrix(hps, 9, &matlfTransform, TRANSFORM_REPLACE);

GpiMove(hps, &ptlStart);                /* Move to starting point       */
GpiPolyLine(hps, sizeof(aptlTriangle) / sizeof(POINTL), aptlTriangle);
```

Shearing a Picture

The following figure is an example of shearing a picture by modifying the transformation matrix directly.

```
MATRIXLF matlfTransform;
POINTL ptlStart, ptlEnd;

ptlStart.x = 500;          /* x coordinate, lower-left corner of box   */
ptlStart.y = 300;          /* y coordinate, lower-left corner of box   */
GpiMove(hps, &ptlStart);
ptlEnd.x = 700;      ;     /* x coordinate, upper-right corner of box  */
ptlEnd.y = 500;      ;     /* y coordinate, upper-right corner of box  */
GpiBox(hps, DRO_OUTLINE, &ptlEnd, 0, 0);  /* Draw first box           */

 matlfTransform.fxM11 = MAKEFIXED(1, 0);
 matlfTransform.fxM12 = MAKEFIXED(0, 0);
 matlfTransform.lM13 = 0;
```

```
    matlfTransform.fxM21 = MAKEFIXED(0, 65536 / 2);  /* Shear factor .5   */
    matlfTransform.fxM22 = MAKEFIXED(1, 0);
    matlfTransform.lM23 = 0;
    matlfTransform.lM31 = 200;              /* Translate 200 units right */
    matlfTransform.lM32 = 0;
    matlfTransform.lM33 = 1;
    GpiSetDefaultViewMatrix(hps, 9, &matlfTransform, TRANSFORM_REPLACE);

    GpiMove(hps, &ptlStart);
    GpiBox(hps, DRO_OUTLINE, &ptlEnd, 0, 0); /* Draw sheared box         */
```

Using World to Model Space Transformations

The following is an example of a sequence of calls in which segment, model, and instance transformations are applied to graphics objects.

```
GpiSetDrawingMode (DM_RETAIN)   /* Sets the current drawing mode        */
                                /* to DM_RETAIN                         */

GpiOpenSegment (segment 1)      /* Creates a chained segment            */

GpiCloseSegment
GpiSetSegmentAttrs              /* Make segment 1 an unchained segment  */

GpiOpenSegment (segment 2)      /* Creates a retained, chained segment  */

GpiSetModelTransformMatrix (TRANSFORM_ADD)
                                /* Specifies a transformation to        */
                                /* apply to subsequent primitives.      */
                                /* This is in addition to the current   */
                                /* model transformation.                */

GpiCallSegmentMatrix (1, TRANSFORM_ADD)
                                /* Calls segment 1 and applies a        */
                                /* transformation to it.                */
                                /* This transformation is in addition   */
                                /* to the current model transformation, */
                                /* and applies only to the called       */
                                /* segment.                             */
   .
   .
GpiSetCurrentArcParams
GpiPointArc                     /* The 3-point arc is not subject       */
                                /* to the transformation specified      */
                                /* on the call to GpiCallSegmentMatrix. */
                                /* The transformation that was          */
                                /* current before segment 1 was called  */
                                /* is applied to the remainder of       */
                                /* segment 2.                           */

GpiCloseSegment
GpiSetSegmentTransformMatrix (segment 2)
                                /* Specifies a segment transformation   */
                                /* for segment 2                        */

GpiDrawSegment                  /* Draws segment 2                      */
```

Viewing Transformation

Each time an application draws the model space, it specifies a different viewing transformation to transfer a view of the picture-to-page coordinate space. GpiSetViewingTransformMatrix is used to apply a viewing transformation. There is one viewing transformation for each part or whole model space to be incorporated in page-coordinate space.

GpiSetViewingTransformMatrix cannot be called while there is an open segment, and it has no effect on primitives outside segments. When it has been specified, the viewing transformation applies to all subsequently created segments until it is next changed. The viewing transformation that is current when a segment is created is a fixed part of the segment. Once specified, the viewing transformation cannot be queried, nor can it be altered, unless you re-create the segment. This is inconvenient when several versions of the same picture are being drawn. The problem can be solved by:

- Defining the picture in one or more unchained segments.
- Creating a segment chain and calling the unchained segments from each root segment.
- Setting the viewing transformation before each root segment.

Each time the segment chain is drawn, multiple views of the same picture are produced. This sequence is illustrated in the following example, where the segment chain comprises three root segments, each of which calls a single unchained segment.

```
GpiSetInitialSegmentAttrs      /* Switches off the chained attribute   */

GpiOpenSegment                 /* Creates an unchained segment         */
                               /* containing the picture definition    */

GpiCloseSegment
GpiSetInitialSegmentAttrs      /* Switches on the chained attribute    */

GpiSetViewingTransformMatrix   /* Sets the viewing transformation      */
                               /* for segment 1                        */

GpiOpenSegment (segment 1)
GpiCallSegmentMatrix           /* Calls the unchained segment          */

GpiCloseSegment
GpiSetViewingTransformMatrix   /* Sets the viewing transformation      */
                               /* for segment 2                        */

GpiOpenSegment (segment 2)
GpiSetViewingLimits            /* Specifies the area of interest       */
                               /* in the model space.                  */

GpiCallSegmentMatrix           /* Calls the unchained segment          */

GpiCloseSegment
GpiSetViewingTransformMatrix   /* Sets the viewing transformation      */
                               /* for segment 3                        */

GpiOpenSegment (segment 3)
GpiCallSegmentMatrix           /* Calls the unchained segment          */
GpiCloseSegment
```

When a segment chain has been created using this method, an application cannot change the viewing transformation unless it re-creates the segment chain. The viewing transformation of a segment is permanently recorded and cannot be edited. The application would not, however, have to re-create the picture definition in the unchained segment.

If the picture definition comprises a number of unchained segments, an application must create an intermediate segment to contain the GpiCallSegmentMatrix calls for those segments. Each root segment would then call the intermediate segment.

The viewing transformation applies to the entire root segment and cannot be overridden from within the segment. It is particularly useful for positioning and scaling one or more segments of a subpicture within the presentation page when a segment transformation cannot be used. A segment transformation can be overridden by any model or instance transformations within the segment. A typical example of its use is when importing a subpicture using GpiPutData or GpiPlayMetaFile.

Note: The viewing transformation must be set to its default value before an application defines an unchained segment to be called from another segment.

Editing Retained Graphics and Graphics Segments

This chapter describes editing retained graphics and graphics segments. The following topics are related to the information in this chapter:

- Presentation spaces and device contexts
- Coordinate spaces and transformations
- Creating and drawing retained graphics

About Editing Retained Graphics and Graphics Segments

In the OS/2 operating system, applications store retained graphics in *segments*. One of the advantages of using graphics segments is that segments can be edited, which allows the retained image to be modified without having to re-create the unmodified portion with multiple GPI functions.

Creating and Drawing Retained Graphics described how to create a segment, and store GPI functions within the *segment bracket* (the GPI functions bracketed by GpiOpenSegment and GpiCloseSegment). To understand how to edit a segment, the underlying structure of a segment is described here in greater detail.

The GPI functions are not inserted directly into a segment bracket. Instead, the operating system converts the GPI functions into *graphics orders*. A graphics order, also known as a drawing order, is the smallest complete portion of a segment. Once the GPI functions are converted into graphics orders, these orders are stored in the GpiOpenSegment-GpiCloseSegment bracket.

Generally, each of the GPI functions within the segment bracket generates one *element* of the segment. An element is the smallest portion of a segment that can be edited and is made up of one or more orders. The following figure illustrates the levels of graphic segment construction.

```
GpiOpenSegment (...);    Graphic Segment
                                                    element
GpiLine (...);            line order             element 0  ~ pointer

GpiBeginElement (...);    begin element order    element 1

   GpiBox (...);               box order
   GpiPolyline (...);          polyline order

GpiEndElement (...);      end element order

GpiLabel (...);           label order            element 2

GpiSetColor (...);        set-color order        element 3

GpiCloseSegment (...);
```

Structure of a Graphics Segment

Elements 0, 2, and 3 are each composed of a single order. Element 1 is composed of two orders bracketed by GpiBeginElement and GpiEndElement.

Graphics Orders

A graphics order is a low-level graphics command that corresponds to a primitive function or attribute. In addition to code and data requirements, each graphics order uses approximately 11 bytes of storage. An application that uses 2000 graphics drawing orders will use around 22KB of memory to store them. The following table describes the four sizes of graphics orders.

Graphics Orders

Graphics Order Size	Content
1 byte	A hexadecimal identifier corresponding to a drawing function or attribute function. This identifier is also known as the order code.
2 byte	The order code is in the first byte, and data is in the second byte.
Long	The order code is in the first byte. The length value of the actual data, in bytes, is in the second byte. The actual data (up to 255 bytes).
Very long (maximum length of 64KB)	A hexadecimal identifier specifically for extended orders, is in the first byte. The order code is in the second byte. A length value that specifies how many bytes are used by the graphics-order arguments, (high order) is in the third byte. A length value that specifies how many bytes are used by the graphics-order arguments, (low order) is in the fourth byte. The actual data.

The following example shows a long graphics order that corresponds to GpiLine:

```
81 8 100 0 0 0 100 0 0 0
```

The first number, 81, is the hexadecimal identifier that corresponds to GpiLine. The second number, 8, is the length value that specifies how many bytes are used by the graphics-order arguments. The next eight bytes contain the arguments for GpiLine. In this case, the arguments specify the line's end point at (100,100).

```
                          Graphics
                          Segment

                           0x21            order #
                           0x08            length
ptl.x = 100L;              0x00
ptl.y = 100L;              0x00               x
GpiMove (hps, &ptl);       0x10                          element
ptl.x = 200L;              0x00
ptl.y = 400L;              0x00
GpiMove (hps, &ptl);       0x00               y
                           0x20
                           0x00
                           0x81            order #
                           0x08            length
                           0x00
                           0x00               x
                           0x30                          element
                           0x00
                           0x00
                           0x00               y
                           0x40
                           0x00
```

The encoding that appears in the Graphics Segments column is a hexadecimal version of the order, length, and parameter information.

In most cases, graphics orders in a segment correspond to a subpicture, which is part of a complete, more complex picture. Your application would combine the individual segments to form the complete picture.

Three drawing modes affect how the operating system stores graphics orders in segments. These modes are described in the following table. The default drawing mode is draw (DM_DRAW). To specify another as the current drawing mode, use GpiSetDrawingMode.

The *actual drawing mode* is a combination of the drawing mode as set using GpiSetDrawingMode, and the segment status-chained, unchained, or outside of the segment. The actual drawing mode does not affect the storing of orders in segments.

Segment Graphics Drawing Modes

Drawing Mode	GpiSetDrawingMode Value	When this mode is set...
Draw	DM_DRAW	it is not possible to store graphics orders in a chained segment.
Retain	DM_RETAIN	your application can store graphics orders in chained and unchained segments.
Draw-and-retain	DM_DRAWANDRETAIN	your application can store graphics orders in chained and unchained segments. In this mode, output intended for a chained segment is both drawn on the device and stored in a segment.

Graphics Elements

Graphics elements are composed of either a single graphics order, or a series of graphic orders that are bracketed by an opening and closing element function.

The purpose of this element bracket is to allow the addition of a single element, that comprises more than one graphics order, to a segment. This is most useful when you know that you will want to retrieve those orders at a later time, and manipulate them as a group rather than as individual functions.

Note: There is no reason for enclosing a single GPI function within a GpiBeginElement-GpiEndElement bracket, although it is a valid construction.

Element brackets also are valid when drawing graphics directly to the output device, but again, they serve no purpose unless converted to a retained segment at a later time. Elements cannot be nested, so two elements cannot begin in succession without an intervening GpiEndElement request. Certain GPI functions cannot be called between GpiBeginElement and GpiEndElement. For example, GpiLabel cannot be used between GpiBeginElement and GpiEndElement.

Element construction is valid only within segments.

Adding Elements to a New Segment

There are four ways of adding elements to a newly created segment:

- Call one or more GPI functions. Each function generates one element of the segment. This is the simplest way of supplying graphics data to a segment.
- Call GpiBeginElement, and follow it with one or more GPI functions. Close the element using GpiEndElement. All the GPI functions enclosed within the bracket and GpiBeginElement and GpiEndElement generate a single element of the segment, and therefore occupy a single-element-pointer position. Subsequently, the graphics orders cannot be accessed individually with the element pointer, although the orders can be accessed directly, for example, using GpiGetData.
- Call GpiElement. As input, you supply the graphics orders themselves, rather than the GPI functions that generate them. No matter how much data you supply on this request, it is considered a single element of the segment.

 Multiple orders added to a segment in this way cannot be accessed individually using the element pointer, but they can be accessed directly, for example, using GpiGetData.

 GpiElement cannot be included within a GpiBeginElement-GpiEndElement bracket. The data passed by GpiElement must not include GpiBeginElement and GpiEndElement.
- Call GpiPutData. As with GpiElement, supply graphics orders as input. You can even include GpiBeginElement and GpiEndElement orders in this data, and thus add more than one element to the segment. GpiPutData is useful when there is a large amount of data to add to a segment.

 GpiPutData cannot be used if the segment editing mode is set to SEGEM_REPLACE. When the segment editing mode is SEGEM_INSERT, the element pointer must point to the last element in the current segment. The segment editing mode can be set by using GpiSetEditMode..

 GpiPutData is most often used in conjunction with GpiGetData, which copies data as a list of drawing orders from a segment to application storage.

Element Types

Each element that results from a single GPI function has an associated *element type* .

The element type is generally the graphics-order code associated with the graphics order generated by the GPI function, and is supplied by the system. For example, GpiLine has the element type, OCODE_GCLINE, which identifies it as a line-drawing request. Your application can determine the current classification of an element using GpiQueryElementType.

When using GpiElement or GpiBeginElement, you can simultaneously add several graphics orders to a segment. Those orders become one element of the segment. You can supply your own classification for these compound elements by supplying a value for the type parameter of GpiElement or GpiBeginElement in the range of 0x81000000 through 0XFFFFFFFF.

For example, the orders passed by a single GpiElement function could change the line type to dotted, set the current color to blue, and draw a line. You could classify this element with a type value that identifies it as a blue dotted line.

Note: Element types are used only for classification. They are not used as location markers or correlation tags.

Element Pointer

Use the *element pointer* to change the contents of an existing segment. When calling GpiOpenSegment, the element pointer is set to 0. The first element that you add to a newly created segment causes the pointer to be set to 1, and each subsequent element causes the pointer to be incremented by 1. When a segment is closed, the element pointer is always reset to 0.

Your application can move the element pointer to a specified value using GpiSetElementPointer. For example, to address the sixth element in a segment, set the element pointer to 6.

Your application also can move the element pointer by a specific number, rather than to a specific number, using GpiOffsetElementPointer. For example, if the element pointer currently addresses element 3, to move it to element 8, code an offset of 5 in GpiOffsetElementPointer. The pointer moves backward rather than forward through the segment. The offset value can be a positive or negative number.

If the sum of the offset and the current pointer position is less than 0, the operating system sets the pointer so that it points to position 0. Position 0 precedes the first element in the segment. If the sum of the offset and the current pointer position is greater than the number of elements in the segment, the operating system sets the pointer so that it points to the last element in the segment.

You can determine the current location of the element pointer using GpiQueryElementPointer. To request the contents of a part of or all of the elements at the current pointer position, use GpiQueryElement.

Labels

You can include *labels* in a segment to make locating particular elements of the segment easier. If you label an element that is likely to be edited, you can access that element when necessary, regardless of whether the actual position of the label-element pair within the segment has changed.

Warning: When editing segments, do not inadvertently insert elements between a label and the element that follows. Your application will no longer be able to use certain GPIs correctly because their default locations and offsets will be invalid.

A label is only a place-holder or reference point, and is itself an element of the segment. Labels are long integer values. To include a label in a segment, use GpiLabel. The following example includes the label 5 in the current segment, whose segment id is 4.

```
GpiOpenSegment(hps, 4L);
GpiLabel(hps, 5L);
 .
 .
 .
```

Your application can set the element pointer to the label position using GpiSetElementPointerAtLabel, then increment the pointer position to the element itself using GpiOffsetElementPointer with an offset of 1.

Note: This example of GPI cannot work properly if your application has allowed the insertion of elements between a label and the element that follows it.

Labels do not have to be unique. If you do not use unique labels, GpiSetElementPointerAtLabel positions the element pointer at the first occurrence of the label, starting from the current pointer position. If the label is not found between the current pointer position and the end of the segment, an error condition is raised. Using labels as a navigation aid might be quicker than scanning the segment for a particular element.

If you do not use labels, you must locate an element before you can change it. To locate a specific element in a segment, you must repeatedly move the element pointer using GpiSetElementPointer then read the contents of the element using GpiQueryElement. This can be quite a lengthy procedure depending upon the number of elements in the segment.

Graphics Segments

Segments are made up of elements, which are in turn made up of one or more graphic orders. The previous sections have discussed the properties and functions for orders and elements. The following sections discuss the attributes and functions that apply to segments as a whole.

The segment attributes are encoded in a segment prior to the graphics orders that correspond to GPIs and attributes.

Segment Attributes

Each segment has a number of characteristics, called *attributes* that you can set in accordance with your application's requirements. The attributes of a segment are quite different from those of a graphics primitive, in that segment attributes have ON and OFF settings. There are seven segment attributes, each described in the following table.

Graphic Segment Attributes and Initial Settings

Attribute	Value	If set...	Default
Chained	ATTR_CHAINED	the operating system adds each new segment in your application's presentation space to the segment chain.	ON
Fast chain	ATTR_FASTCHAIN	the operating system prevents the resetting of primitive attributes to their default values before drawing the segment chain.	ON
Dynamic	ATTR_DYNAMIC	the operating system draws segment output by using the XOR raster operation.	OFF
Detectable	ATTR_DETECTABLE	your application can perform correlation operations on segments created in this presentation space.	OFF
Propagate detectable	ATTR_PROP_DETECTABLE	the detectable attribute will be set in each segment called by a chained segment.	ON
Visible	ATTR_VISIBLE	GpiDrawChain, GpiDrawFrom, and GpiDrawSegment will generate output on a device.	ON
Propagate visible	ATTR_PROP_VISIBLE	The visible attribute will be set in each segment called by a chained segment.	ON

When an application creates a segment in a presentation space, the operating system assigns initial attributes to it. By default, five of the attributes will be set ON and two will be set OFF. Your application can override the defaults for all subsequently created segments within the presentation space using GpiSetInitialSegmentAttrs.

Use GpiQueryInitialSegmentAttrs, to retrieve the values of the current initial attributes.

The Dynamic Attribute

Dynamic segments are graphics segments that can be moved from where they were drawn on the screen to a different coordinate position, or altered in some other way, without affecting the remaining part of the picture. The new position of a dynamic segment can be provided with an input device, such as a mouse, or it can be application-generated.

The dynamic attribute is set to OFF by default. The setting is always inherited by called segments from the setting of their callers. If a calling segment is nondynamic, any segments called by it are also nondynamic, regardless of how they have been defined. If the calling segment is dynamic, all segments called by it are also dynamic. Any segment explicitly defined as dynamic must have a unique name, and it cannot be created when the current drawing-mode parameter is DM_DRAW or DM_DRAWANDRETAIN. Although no error condition is raised if you define an unchained segment as dynamic, it is not treated as dynamic unless the segment is called from a dynamic root segment.

Graphics objects to be moved without disturbing the remaining contents of the display are drawn in exclusive-OR mode. Segments that you define as dynamic always are drawn in exclusive-OR mode, regardless of the current mix attribute settings and any GpiSetMix functions the segment itself might contain.

For performance reasons, dynamic segments are to be grouped at the start of the picture chain. There are GPI functions that handle dynamic segments as a group, so it is more efficient if the entire segment chain does not have to be scanned to locate them.

When the entire picture chain is drawn, however, dynamic segments are to be drawn *after* all other segments in the chain to ensure that the effects of drawing in exclusive-OR mode are not adversely affected by drawings in other mix modes. Also, you must ensure that no nondynamic drawing overlays the dynamic segments after they have been drawn. The PM ensures that dynamic segments are always drawn on top of other graphics.

The Detectability Attribute

The detectability attribute of a segment determines if that segment can be selected with an input device, such as a mouse. A segment with this attribute is said to be *selectable* . Selectable segments can be selected for correlation operations. The detectability attribute is set OFF by default. All detectable segments must have unique names. If you define a zero segment as detectable, it is created as nondetectable.

The Propagate-Detectability Attribute

This attribute controls whether the detectability attribute of a calling segment is inherited by any segments called from it. The propagate-detectability attribute is set ON by default. This value overrides the detectability setting of the called segment. For example, if you set the detectability attribute of a calling segment to OFF, all segments called from it are nondetectable, regardless of how they have been defined.

The Visibility Attribute

The visibility attribute controls whether a segment is to be visible on an output device. The contents of a segment that is not defined with the visibility attribute set ON are still executed whenever the segment is drawn. This can cause changes to the current position and to current attribute values, even though the primitives themselves are not visible on the output device. The visibility attribute is set ON by default.

The Propagate-Visibility Attribute

This attribute controls whether the visibility attribute of a calling segment is inherited by any segments called from it. The propagate-visibility attribute is set ON by default. This value overrides the visibility setting of the called segment. For example, if the visibility attribute of a calling segment is set to OFF, all segments called from it are invisible, regardless of how they have been defined.

Changing the Attributes of a Segment

After you create a segment, you might need to alter its attributes. For example, if you created a segment using the default attributes and you want to perform a correlation operation on the subpicture in that segment, you will need to set the detectable attribute using GpiSetSegmentAttrs. You can retrieve the values of the attributes for any segment using GpiQuerySegmentAttrs.

You can change the default segment-attribute settings globally for a single presentation space using GpiSetInitialSegmentAttrs. The attribute setting that you specify in this function applies to all segments created subsequently in that presentation space, except that a nonretained segment can never have the dynamic attribute. For example, if you want all segments to be detectable, you can call this function to change the setting for all of them *before* you create them. GpiSetInitialSegmentAttrs cannot be used to change the attributes of existing segments.

You also can change the attributes of any single retained segment, but not those created subsequently, using GpiSetSegmentAttrs. This is useful if, for example, you want most of the segments in a picture chain to be detectable. You can change the attribute setting to detectable for the entire chain, before creating it, using GpiSetInitialSegmentAttrs; then, change the attribute to nondetectable for each of the segment exceptions using GpiSetSegmentAttrs. You can also use GpiSetSegmentAttrs, for example, to make a visible segment invisible when an erase request is received from the operator, or to take a segment out of the chain by redefining it as unchained. If you change an unchained segment to chained, it is added to the end of the segment chain. If you want the newly chained segment to be positioned elsewhere in the chain, use GpiSetSegmentPriority rather than GpiSetSegmentAttrs.

Some of these segment attributes apply equally to primitives that are outside of segments, although their default settings cannot be changed. Primitives outside of segments are always detectable and visible. They cannot be dynamic, because the dynamic attribute applies only to retained graphics. The chained and fast-chaining attributes do not apply to primitives outside of segments.

Editing a Segment

The operating system provides segment editing functions for writing applications that allow users to edit segments or elements in a segment. For example, after performing a correlation operation using your application, a user might need to alter the elements that intersected the pick aperture. Correlation and the pick aperture are explained in in Correlation.

You can edit the contents of any retained segment that has a unique name. You also can edit the contents of a retained zero segment that has not yet been closed. Zero segments cannot be edited after they have been closed, because you cannot refer to a zero segment when it is no longer the current open segment.

Before you can begin editing, you must set the current drawing-mode parameter to DM_RETAIN. You cannot edit a segment if the current drawing-mode parameter is DM_DRAWANDRETAIN or DM_DRAW. To begin editing, call GpiOpenSegment and specify the name of the segment you want to edit. When you are finished editing a segment, close it using GpiCloseSegment.

The operating system has two edit modes: *insert* mode and *replace* mode. You can set the edit modes using GpiSetEditMode. You can determine which mode is currently set using GpiQueryEditMode.

The current edit mode applies to all segments in the presentation space until you change it. The edit mode is not an attribute of a particular segment and can be changed at any time. The default edit mode, which is set when you create a presentation space, is insert mode. When you create a graphics segment, you are actually editing it in insert mode.

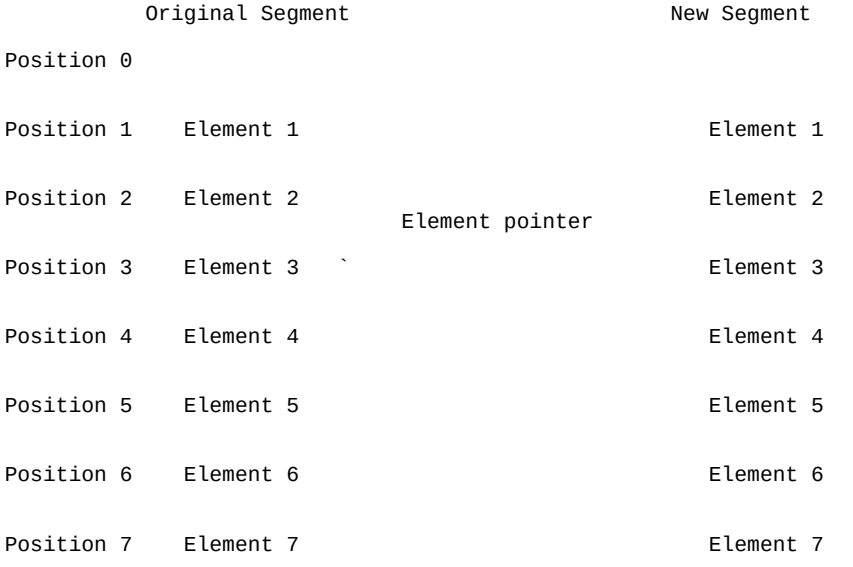
If the edit mode is set to insert (SEGEM_INSERT) you can insert an element at the current location of the element pointer. The operating system shifts the element that was previously at that location into the next slot, and so on, until the last element is shifted into a new, final slot. The following figure shows a segment before and after a new element is inserted at position 0, the beginning of the segment.



Inserting a New Element in a Segment

The new element is inserted *after* the current element; then, the element pointer is set to the new element.

If replace mode is set (SEGEM_REPLACE) you can replace the element at the current pointer location with a new element. The following figure shows a segment before and after the third element was replaced.



Replacing an Element with a New Element

The new element overwrites the previous element; the element pointer does not change.

Replacing elements is the recommended technique for redrawing identical primitives with different attributes-for example, color.

You can insert or replace data in an existing segment using any of the functions described in Adding Elements to a New Segment. As mentioned previously, GpiPutData is valid only in insert mode and only when the element pointer is addressing the final element in the segment. That is, when you are editing, you can use GpiPutData only to add data to the end of a segment.

A replace request is not valid when the element pointer is set to 0. If you call GpiOpenSegment to create a new segment without first ensuring that the current edit mode is insert, your first attempt to add an element to the segment might cause an error condition to be raised.

Deleting a Graphics Element

The PM has three different functions that allow you to delete elements within the currently opened segment.

Function	Description
GpiDeleteElement	Deletes a single element at the current element pointer location.
GpiDeleteElementRange	Deletes a range of elements in a segment.
GpiDeleteElementsBetweenLabels	Deletes a series of elements between two labels.

When you remove an element from a segment, the element pointer addresses the element immediately before the one you deleted. As with other editing functions, the current drawing mode parameter must be DM_RETAIN when you are deleting elements. The current editing mode (insert or replace) has no effect on the delete functions.

Deleting Graphics Segments

When you have finished drawing the subpicture associated with a segment, you should delete the segment. The PM has two different functions that allow you to delete segments.

Function	Description
GpiDeleteSegment	Deletes a retained segment
GpiDeleteSegments	Deletes a range of retained segments.

Your application can only delete segments that have unique names because you use the segment identifiers to identify the segment or range of segments for deletion. Retained zero segments cannot be deleted. They disappear only when their associated presentation space is deleted or reset. The presentation space can be reset either by GpiResetPS with the GRES_SEGMENTS flag or the GRES_ALL flag set, or by GpiSetPS. Resetting the presentation space is described in Reusing the Presentation Space.

If GpiDeleteSegment occurs within a segment bracket and the segment identified is the currently open segment, the operating system deletes the segment and ignores the remainder of the functions up to, and including, GpiCloseSegment. If GpiDeleteSegment occurs within a segment bracket and the identifier for a segment other than the currently open segment, the operating system deletes the segment, then continues processing the remaining functions in the segment bracket. If GpiDeleteSegment occurs outside of a segment and references a segment in the segment chain, the operating system removes the segment from the chain and links the two adjacent segments, if such segments exist.

If the range of segments deleted by GpiDeleteSegments were in the segment chain, the operating system "repairs" the chain by linking the

segments immediately preceding and following the deleted segments, if such segments exist.

Copying a Single Graphics Element

Your application can copy the graphics orders from a single element using the GpiQueryElement and GpiElement functions. GpiQueryElement copies the graphics orders (or part of the orders, depending on allocated size) from an element into an array of bytes. GpiQueryElement is not valid within an element bracket, nor if the drawing mode is other than DM_RETAIN.

GpiElement copies the data from the array back into the current segment. The data may not contain any begin or end element orders. GpiElement is not valid within an element bracket. The element is retained if the drawing mode is DM_RETAIN and drawn if the mode is either DM_DRAW, or DM_DRAWANDRETAIN.

Copying Multiple Graphics Elements

Your application can use GpiGetData and GpiPutData to:

- Copy elements from one segment to another
- Copy elements from one position in a segment to another position in the same segment.

GpiGetData copies a buffer of graphics orders from a named segment, into an area of application storage, whose byte size you specify on input. Only the named segment must not be open when you call this function. The named segment is also known as the *source segment* .

The first time GpiGetData is called, data retrieval has to start at the beginning of the segment. This is done by declaring an element offset of 0.

Output from GpiGetData depends on whether the area of application storage is large enough to hold the entire buffer of graphics orders. If so, the buffer is returned along with a count of the number of bytes of data returned to you from the segment. If not, the application storage is filled, and the count is set to the size of the application storage. The offset of the element where GpiGetData stopped copying, and the count are returned. Your application can then determine if a subsequent GpiGetData needs to be called, by checking that the count is less than the size (length) of the application storage.

On subsequent GpiGetData functions, the element offset can be 0, or it can be the offset value that was returned from the previous function. In this manner, if your application calls GpiGetData more than once to copy an entire segment, it can "pick up where it left off", rather than recopying the segment from the beginning each time.

You can copy the data from the application storage to a new location in a segment using GpiPutData. This receiving segment is also known as the destination segment. The segment into which you are copying the data can be open when you call GpiPutData.

If GpiGetData does not retrieve a complete segment, the data it does retrieve can be written out to the second segment using GpiPutData, even if the last order copied is incomplete.

The current drawing mode determines whether the graphics orders are executed, stored in the segment, or both. When you call GpiPutData, the current edit mode must be SEGEM_INSERT and, if there is already data in the destination segment, the element pointer must address the last element of the segment.

Because the PM does not support explicit renaming of segments, GpiGetData-GpiPutData is the method you use to rename a segment. That is, create a second segment with the desired name, copy the contents of the first segment to it, then delete the original segment.

Drawing Retained Graphics

Nonretained graphics segments and primitives outside of graphics segment are always drawn immediately to the current output device. Segments that have been retained in segment store, however, can be drawn any number of times to any number of device contexts. The

PM has three different functions that allow you to draw segments:

Function	Description
GpiDrawSegment	Draws a single segment in the chain
GpiDrawFrom	Draws a group of segments in the chain
GpiDrawChain	Draws the entire chain

GpiDrawSegment draws a single, named segment, which can be chained or unchained. You supply the segment name as input to this function. GpiDrawSegment can draw a dynamic segment in some circumstances. These are described in Drawing Dynamic Segments.

GpiDrawFrom draws one or more root segments from the segment chain. You supply the names of two root segments as input. The drawing process starts at the first named segment and draws all chained and called segments, excluding dynamic segments, up to and including the second named segment.

The order in which you specify the segment names is to reflect their relative positions in the segment chain. If the second-named segment appears in the segment chain before the first-named segment, no error is raised, but drawing starts from the first named segment and continues to the end of the chain. This also happens if the second segment is an unchained segment rather than a root segment. If the drawing control, DCTL_DYNAMIC is set, any dynamic segments already drawn on the display are removed before GpiDrawFrom begins to draw nondynamic segments. Then they are redrawn after GpiDrawFrom ends.

GpiDrawChain draws all nondynamic chained segments in a named presentation space, as well as any unchained segments called by those chained segments. It does not draw dynamic segments. You supply only the presentation space handle as input to this function. If the drawing control, DCTL_DYNAMIC is set, any dynamic segments already drawn on the display are removed before GpiDrawFrom begins to draw nondynamic segments. Then they are redrawn after GpiDrawFrom completes.

You can call any of these functions while a segment is open. The open segment is not modified by these functions, although any of them can result in the current attribute values and current position being modified. For more information about the attribute-resetting process, see Graphics Attributes.

The current drawing mode determines whether the segments identified in the drawing functions are executed, stored in the segment, or both. The current drawing mode does not affect which segments are drawn, or the priority in which they are drawn for any of these three GPI drawing functions.

If an error occurs during the drawing process, you can determine where the error occurred using GpiErrorSegmentData. This function returns information about the last error that occurred during a segment-drawing operation. It returns a pointer to the segment identifier and a pointer to one of three constants that indicate when the error occurred.

Constant	Error occurred...
GPIE_SEGMENT	While drawing the segment
GPIE_ELEMENT	While calling GpiElement
GPIE_DATA	While calling GpiPutData

Drawing Dynamic Segments

A *dynamic segment* is a chained segment that possesses special properties. Dynamic segments are those that can be moved or changed in some other way without disturbing the remaining parts of the picture. To draw dynamic segments on the display screen for the first time, use GpiDrawDynamics. GpiDrawDynamics draws every dynamic segment in the segment chain.

When they have been drawn on the display, dynamic segments can be updated. For example, to move one or more dynamic segments to another part of the display screen, remove them from their current position using GpiRemoveDynamics. You can restrict the scope of this function by supplying the names of a start segment and a finish segment. The start segment and finish segment can be the same. Dynamic segments outside of this range are not removed from the display screen.

If you intend to re-associate the presentation space, call GpiRemoveDynamics first; otherwise, GpiRemoveDynamics cannot remove the dynamic segments after the presentation space has been re-associated. The dynamic segments effectively cease to be dynamic.

After removing the appropriate dynamic segments from the display screen, you can redraw them elsewhere, again using GpiDrawDynamics. If you specify a range of dynamic segments in GpiRemoveDynamics, the redrawing is restricted to that range. Note that, when dynamic segments have been drawn on the display screen, you must not edit their definitions in segment store.

Dynamic segments are treated like nondynamic segments when you are directing output to a hardcopy device. GpiRemoveDynamics and GpiDrawDynamics raise an error condition if the current output device is not a display window.

The Drawing Controls

GpiErase clears the output display of the currently associated device to CLR_BACKGROUND, which is the normal background color for the device. If you are using the default color table, this value clears the window background to white. This is a once-only request and must be issued each time you want the display screen cleared before drawing a new picture. You can use this function for either a micro presentation space or a normal presentation space.

To get this effect for more than one drawing request, you can use GpiSetDrawControl. This function establishes current values for five drawing controls, which remain in effect until they are reset. The following table describes the five drawing controls.

Drawing Controls

Control	Value	If set, the operating system...
Boundary data accumulation	DCTL_BOUNDARY	computes the dimensions of the smallest rectangle that would completely surround the retained-drawing output. This control is described in Clipping and Boundary Determination.
Correlation	DCTL_CORRELATE	performs correlation operations on any primitives or any output associated with GpiPutData or GpiElement. This control is described in Correlation.
Display control	DCTL_DISPLAY	draws retained output on the device identified by the current device context. If this control is not set, no retained output will appear on the device.
Draw-dynamic-segments	DCTL_DYNAMIC	calls GpiRemoveDynamics before drawing any retained output and then, after drawing the retained output, it calls GpiDrawDynamics to draw output stored in dynamic segments. This automatically removes all dynamic segments from the display screen before a GpiDraw... request is issued, then later redraws

		the segments. This ensures that dynamic segments are always drawn on top of nondynamic segments and primitives.
Erase-before-draw	DCTL_ERASE	calls GpiErase before drawing any retained output.

GpiSetDrawControl can be called in either a micro or a normal presentation space, although not all of the controls are valid in a micro presentation space. GpiDrawSegment can be used to draw a dynamic segment in some circumstances. Its effects are as follows:

- If the named segment is both chained and dynamic, and the DCTL_DYNAMIC drawing control is set, *all* dynamic segments in the chain, including the named segment, are drawn as dynamic.
- If the named segment is both chained and dynamic, and the DCTL_DYNAMIC control is not set, the named segment is not drawn.
- If the named segment is unchained and dynamic, it is drawn as a nondynamic segment, regardless of the setting of the DCTL_DYNAMIC control.

If you called GpiRemoveDynamics prior to calling GpiDrawDynamics and you specified a range of dynamic segments, the operating system draws only that range. If you set the DCTL_DYNAMIC control using GpiSetDrawControl, the operating system calls GpiRemoveDynamics before drawing the subpictures from the dynamic segments.

The DCTL_DISPLAY control is the only control set to DCTL_ON by default. All other controls are set to DCTL_OFF when you create a presentation space.

Using Segment Editing Functions

You can use editing functions to:

- Create a chained-dynamic segment
- Delete a segment
- Edit the contents of a segment

The following figure shows an example of how to edit the contents of a segment. In the example, three elements are inserted. The steps required for this task are as follows:

1. Set the segment edit mode to insert or replace using GpiSetEditMode.
2. Set the drawing mode to retain.
3. Open the segment using GpiOpenSegment, passing it the segment identifier from a previous correlation operation.
4. Set the element pointer so that it points to the position at which you will replace or insert an element using GpiSetElementPointer, GpiSetElementPointerAtLabel or GpiOffsetElementPointer.
5. Insert the new primitives using any of the Gpi primitive functions.
6. Delete any unnecessary primitives using GpiDeleteElement or GpiDeleteElementRange.
7. Close the segment using GpiCloseSegment.

```
#define INCL_GPISEGEDITING
#define INCL_GPICONTROL
#include <os2.h>
void fncESEG01(void){
    HPS hps;
    LONG idNonChained;
    POINTL ptl;
```

```
    GpiSetEditMode(hps, SEGEM_INSERT);

    GpiSetDrawingMode(hps, DM_RETAIN);

    GpiOpenSegment(hps, idNonChained);

    GpiSetElementPointer(hps, 0L);

    GpiSetColor(hps, CLR_YELLOW);

    ptl.x = 100; ptl.y = 100;

    GpiMove(hps, &ptl);

    ptl.x = 150; ptl.y = 150;

    GpiBox(hps, DRO_OUTLINE, &ptl, 40L, 40L);

    ptl.x = 30; ptl.y = 30;

    GpiMove(hps, &ptl);

    GpiSetElementPointer(hps, 5L);

    GpiDeleteElement(hps);

    GpiCloseSegment(hps);
} /* fncESEG01 */
```

The first element inserted contains the graphics order that sets the color to yellow; the second element moves the current position; and the third element draws an outlined box with rounded corners. After the three elements are inserted, the code deletes the element at position 5 in the segment (this element was previously at positions 1 and 2).

Fonts

In typography, a *font* is a collection of characters that share a common height, line weight, and appearance. In Presentation Manager, fonts are an lcid-identified resource, like bit maps, and are used in conjunction with the character string primitive data structures to define the appearance of displayed and printed text. The primary role of the PM with regard to fonts is to evaluate whether a particular font is appropriate for a given situation.

The following topics are related to information in this chapter:

- Presentation spaces and device contexts
- Character string primitives
- Coordinate spaces and transformations
- Color and mix attributes

About Fonts

Font height is specified in printer's points, referred to in this book as just *points*. A point equals approximately 1/72 of an inch. The line weight and appearance of a font are specified by the categories listed in the following table.

Examples of Line Weight and Font Appearance Categories

Attribute	Category	Description
Line Weight	Normal	Characters drawn with a normal line weight

	Bold	Characters drawn with a heavier line weight
Appearance	Condensed	Characters drawn with half the average character width
	Expanded	Characters drawn with twice the average character width
	Italic	Characters drawn with a normal line weight and sloped to the right

A *font family* is a collection of fonts that share common design characteristics such as stroke width and serif. *Stroke width* refers to the width of lines used to draw the characters and symbols of a font. A *serif* is a short cross-line drawn at the ends of the main strokes forming a character or symbol.

Text output is drawn with the characters and symbols of a font. The operating system stores fonts either in memory or on devices. Applications can access fonts through a device context associated with the current presentation space. When an application creates a presentation space, using GpiCreatePS or WinGetPS, the operating system assigns the presentation space a default font from one of the available fonts. An application can retrieve information about the default font using GpiQueryFontMetrics.

Image Font and Outline Font Implementation

The operating system can define and implement the characters of a font as either images or outlines.

Each of the characters in an *image font* (sometimes called a *raster font*) is created by alternating the ON and OFF settings of pels to produce the image. The image characters are stored as a bit map. Sometimes image fonts are even referred to as *bit-map fonts*.

The characters in an image font must be drawn in a fixed size-you cannot rotate or scale them, for example-but usually they are of good quality. They are displayed faster than outline characters, and frequently they look better at low resolutions.

The characters in an *outline font*, on the other hand, are drawn using a sequence of lines and arcs. Then the characters are filled if so requested. The outline characters are stored as a collection of line, fillet, and spline functions. Outline fonts are transformable because the outline characters can be scaled, rotated, and sheared. Outline fonts can be implemented as vectors, drawn by a series of small, straight lines. If that is the case, they can be referred to correctly as *vector fonts*. Vector fonts can be drawn much faster than fonts requiring the inclusion of actual curves and arcs.

A major advantage of outline fonts is their device independence. Unlike image fonts, whose appearance depends on the device's pel definition.

The following figure is an example of a character in both an image font and an outline font.

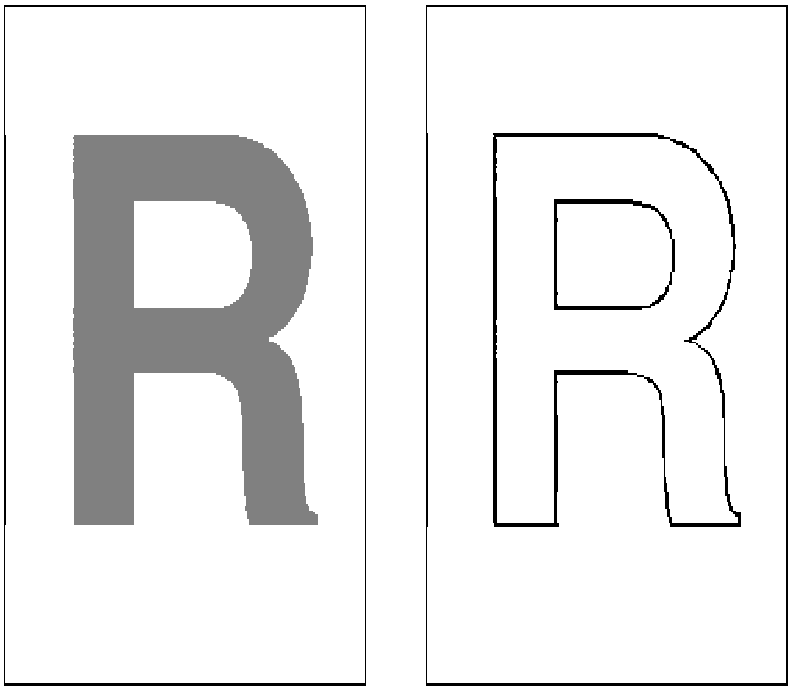


Image and Outline Characters

The image character is produced by manipulating pels. The outline character is produced by drawing a sequence of lines and arcs. (The resultant shape of an outline character can be filled if desired.)

Most of the functions and data structures described in this chapter relate only to outline fonts.

PM-Supplied Fonts

The PM provides a number of both image and outline typographic-quality character fonts. On some devices, image fonts in small font sizes have a better appearance than their outline-font equivalents.

Type size is measured in points. There are approximately 72 points to an inch, so each character in a 24-point font, for example, is 1/3-inch high. Each incremental unit of a presentation page in PU_TWIPS format is 1/20 of a printer's point (1/1440 inch).

The following image fonts are available on all OS/2-supported display adapters:

Font Family Name	Point Sizes Available
Tms Rmn	8, 10, 12, 14, 18, 24
Helv	8, 10, 12, 14, 18, 24
Courier (monospace)	8, 10, 12
System Monospace	10

In addition, a default system font is used in window components such as title bars and menus. It is a proportionally spaced Swiss font.

The outline fonts provided by the PM are Adobe** Type 1. The appearance and performance characteristics of these fonts are more flexible than for image fonts.

The following table lists the outline fonts available with the operating system and their equivalent fonts from earlier versions of the OS/2 operating system.

Available Outline Fonts

Outline Font Family Name	OS/2 Fonts Available	Equivalent Name
Times New Roman**	Times New Roman	Tms Rmn
	Times New Roman Bold	Tms Rmn Bold
	Times New Roman Bold Italic	Tms Rmn Bold Italic
	Times New Roman Italic	Tms Rmn Italic
Helvetica**	Helvetica	Helv
	Helvetica Bold	Helv Bold
	Helvetica Bold Italic	Helv Bold Italic
	Helvetica Italic.	Helv Italic
Courier	Courier	Courier
	Courier Bold	Courier Bold
	Courier Bold Italic	Courier Bold Italic
	Courier Italic	Courier Italic
Symbol	Symbol	None

The outline font names provided with earlier versions of the OS/2 operating system are still supported, but the corresponding new fonts are obtained when the previous font names are selected.

Availability of Additional Fonts

Over 600 Adobe Type 1 fonts are available from font vendors, and user systems could have a large number of these installed. Applications must be capable of functioning properly with whatever font the user selects.

An application can examine the font metrics to see what a font looks like or display different font characters from which the user can choose. When dealing with an interactive or user-driven application, the recommended method of choosing a font is to call WinFontDlg, which displays an example of the font in a dialog box.

Additional Adobe Type 1 fonts can be used just like other outline fonts. Notice, however, that some of those fonts are more *stylized*; that is, the fonts have a greater variation in the widths of different characters within the same font, which is provided by *kerning*.

Adobe Type 1 fonts, provided with and for a particular application, can be loaded with GpiLoadFonts, provided that the following rules are observed:

- The .AFM file of the font must be specified as the font file.
- The .PFB file must be in the same directory as the .AFM file.

Font Data Structures and Attributes

The attributes of fonts are contained in the FONTMETRICS data structure. The appearance of the actual text is influenced also by the attributes of the individual characters, which can be found in the CHARBUNDLE data structure.

An application can determine whether to use a particular font by examining its *font metrics* , which are the measurements that define the features of that font. The measurements are decided on by a font designer, whose most important criteria might be ensuring that the font is pleasing to the eye.

Unlike the attributes of a character string primitive, the individual font metrics attributes cannot be changed using specific GPI functions. Your application can determine the values of the current logical font attributes by calling GpiQueryFontMetrics, which accepts as input the amount of data to be returned as well as a pointer to the data area. Unlike other GPI calls, GpiQueryFontMetrics does not return the size necessary for all font metrics data. Querying with a **sizeof** operator, instead of calling the query twice, returns the size data.

The following table and text defines and explains the uses of the various font metric attributes.

Attribute	Description
Face name	The full typeface name of a font, such as *Courier Bold Italic*.
Family name	A broader equivalent of the face name (e.g.*Courier*).
Code page	The mapping between a set of codepoints and a set of graphic characters.
Character cell	Controls the height and width of outline font characters.
Character cell ascent	Distance between the baseline to the top of the character cell.
Character cell descent	Distance between the baseline and the bottom of the character cell.
Maximum baseline extent	Maximum vertical extent of font characters in world coordinates.
Maximum ascender	Maximum distance a font character ascends above the baseline.
Maximum descender	Maximum distance a font charter descends below the baseline.
Internal leading	Vertical distance equal to the difference between the maximum baseline extent and the em height.
Em height	Equivalent to the character cell height.
Em width	Equivalent to the character cell width.
X height	Height above the baseline of any lowercase character.
Lowercase ascent	Maximum distance of a lowercase character above the

	baseline.
Lowercase descent	Maximum distance of a lowercase character below the baseline.
External leading	Maximum vertical font spacing that can be added without adversely affecting the appearance of the text.
Average character width	Average width of font characters in world coordinates
Character slope	Initial slope of a character with respect to a vertical line.
Inline direction	Character angle measured with respect to a horizontal line.
Weight class	Specifies the thickness of each stroke of a font character.
First, last, default, and break characters	See the topic Glyphs, Code Pages and Code Points.

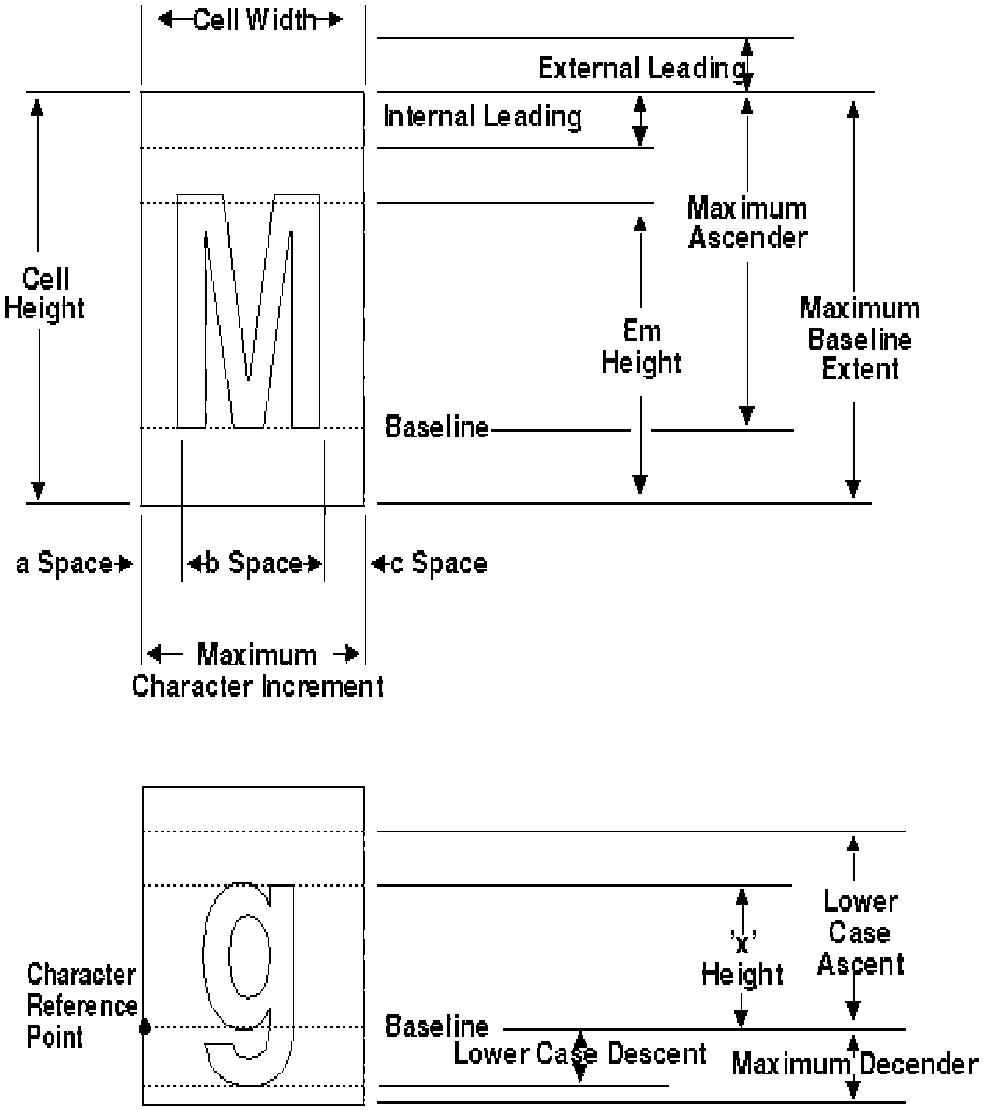
The *face name* attribute is the typeface name by which a particular font design is known. Two examples are Times New Roman Bold and Times New Roman Italic. Roman is the *family name* of these fonts. You always should provide a face name because most outline fonts are known by it.

The *code page* to be associated with the font is identified by this metric. Some fonts are specific to a particular code page and should be used with only that code page. Other fonts are universal and can be used with any supported code page.

Every character in a font is drawn within a rectangular region called a *character cell*. The character-cell height is the distance from the bottom of the character cell to its top. The width of the character cell is the distance from one side to the other.

An imaginary horizontal line, called a *baseline*, is drawn through the lower portion of the character cell. The operating system uses the baseline to position characters. All uppercase and most lowercase letters in a given font rest approximately on the baseline. Some lowercase letters, such as *g* and *y*, descend below the baseline.

The distance from the baseline to the top of the character cell is the *character-cell ascent*. Similarly, the *character-cell descent* is the distance from the baseline to the bottom of the character cell. The following figure illustrates characters drawn along a baseline inside their character cells. Various font metrics also are illustrated.



Some Font Metrics

The *maximum baseline extent* is the sum of the *maximum ascender* and the *maximum descender* values. The *internal leading* is the space allowed for diacritics above capital letters. When an application draws a string of text, the operating system positions the leftmost point of the baseline over a predetermined point for each character in the string.

A font's *Em height* is a measure of its visual height. This measurement was given its name because, historically, the height of an uppercase letter M usually was equal to the average height of all uppercase characters in the font. However, this is no longer the case.

The value of Em height represents the font point size in world coordinates and is the same as the character cell height. For an outline font, this can be set by the character cell height attribute. The value of *Em Width* is the equivalent horizontal dimension and is the same as the character cell width. For an outline font this also can be set by the character cell width attribute. These are, in fact, the dimensions of the *em square* (a typographic term) and, like point size, cannot be defined in terms of any measurable characteristic of a visible character of the font.

The average distance from the baseline to the top of any lowercase character is called a font's *x height*. This measurement was given its name because the height of a lowercase *x* usually is equal to the average height of all lowercase characters in the font.

The *maximum ascender* is the height of the tallest character in a font. The *maximum descender* is the depth (below the baseline) of the lowest character in a font.

The *lowercase ascent* is the height of the tallest lowercase character in a font. The *lowercase descent* is the depth (below the baseline) of the lowest lowercase character in a font.

Many fonts reserve part of the space in the top of each character cell for accent marks. This reserved space is called *internal leading*. Some fonts require a certain amount of space to be left between rows of text. This space, called *external leading*, is not included in the

character-cell height or ascent measurements. The external leading is the recommended space to leave between rows of text to preserve its pleasing appearance.

The *average character width* is stored in the **lAveCharWidth** field in the FONTMETRICS structure. The average character width is a measure of character width based on the normal frequency of lowercase letter usage in the English language. You can use this value to approximate the average width of a character in string length calculations.

The *maximum baseline extent* for a font is the sum of the maximum ascender and the maximum descender. Maximum baseline extent is not equal to cell height for outline fonts, but is for image fonts.

The *character slope* is an angle measured clockwise with respect to a typical vertical line. The slope of a normal font is 0; the slope of an italic font is other than 0. The character slope in the FONTMETRICS data structure can be thought of as the initial slope, since the slope can be changed in the CHARBUNDLE data structure.

The *inline direction* is an angle measured clockwise with respect to a horizontal line. The inline direction for a Swiss, Helvetica, or Roman font is normally 0-exactly horizontal. The inline direction for a Hebrew font is normally 180, because Hebrew text is oriented right to left.

The *character-rotation angle* is an angle measured counterclockwise with respect to a horizontal line. The baselines of characters are aligned with the vector drawn at the angle of rotation.

The weight class specifies the thickness of each stroke that forms part of each character in a font.

The first, last, default, and break character all are associated with glyphs for UGL fonts and code pages.

A *superscript* is a character drawn immediately above and to the right of a normal character in a string of text. Superscripts are identified by a width and height and by vertical and horizontal offsets.

A *subscript* is a character drawn immediately below and to the right of a normal character in a string of text. Subscripts are identified by a width and height and by vertical and horizontal offsets.

Glyphs, Code Pages, and Code Points

The image or picture that you associate with each character or symbol in a font is called a *glyph* . The mapping between a set of glyphs and their *code points* is called a *code page* .

For a single byte character set (SBCS), each code page contains up to 256 code points. Normally these are 8-bit integers in the range 0 through 255, with one code point identifying one glyph in that code page. The code pages can be either ASCII or EBCDIC. Because a code point repeats on a new code page, you need both the code page and code point to uniquely identify a glyph.

There usually is one code page per font. However, double byte character sets (DBCS) are sometimes used for Asian languages with large character sets. Fonts are a set of glyph definitions and a default code page mapping. The same set of font glyph definitions is remapped for different code pages.

Each font contains four special glyph points:

- First glyph
- Last glyph
- Default glyph
- Break glyph

First and last glyph points are of more interest to the font designers than application programmers because they apply to the set of font glyph definitions rather than a particular code page. The last character is the maximum code point of the font that has a glyph associated with it and may be bigger than 256.

The default glyph appears in text when an application specifies a glyph point that does not exist in the font.

The break glyph usually is the space character and often has the same code point as the default character.

Actually a font can have more or less than 256 definitions, but any particular code page may have less than 256 but no more. There may be a translation between the code point in the code page and the code point (between the first and last glyphs) that references the glyph in the font.

The operating system assigns unique identifiers to each of its code pages. Common code pages are 437-the United States code page, and 850-the multilingual code page. The default code page is 850.

An application can determine the current code page by calling GpiQueryCp. or it can assign a new code page using GpiSetCp. If you default the code page in the FATTRS structure when calling GpiCreateLogFont, you get the current code page as specified by GpiSetCp (or

returned by GpiQueryCp). You can specify any one of the code page identifiers returned to you from WinQueryCpList.

When using a font other than the system font, you specify the required code page in the FATTRS structure of GpiCreateLogFont. You can determine the code page for that font (if it is the current logical font) by using GpiQueryFontMetrics.

Proportional and Monospace Fonts

When text is drawn, the operating system aligns each character by positioning its character cell next to that of the previous character.

A *monospace* font provides the same amount of space for each character, whatever its shape. These fonts also are called *fixed fonts*. Courier and System Monospaced are monospace fonts. Monospace fonts, in fact, are essential for some purposes- for example program listings, where vertical alignment is important.

A *proportionally-spaced* font is one in which some characters are allotted less space than others, so that each character occupies the proportion of space that is correct for its shape. For example, a lowercase letter *i* does not need the same space as a lowercase letter *m*. Many graphics fonts supplied by the PM, including the system font, support proportional spacing.

The font metrics value **maxcharinc** specifies the width of the widest character in the font, and the value **avecharwidth** specifies an average width of the characters in the font.

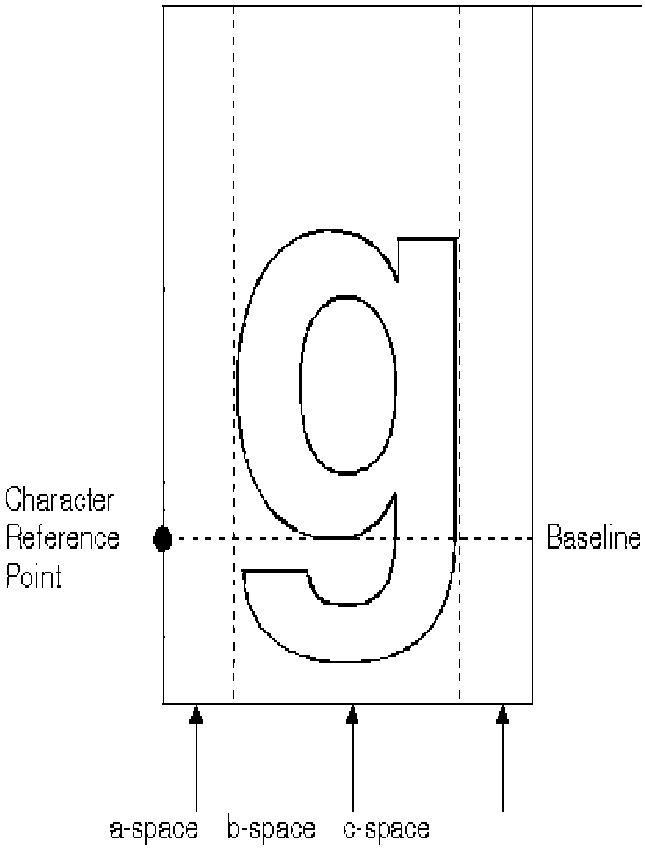
You can retrieve information about the widths of the characters in the current font by calling GpiQueryWidthTable. You provide the code point of the first character you are interested in and the number of characters for which you want width-table information.

Kerning

Kerning, like proportional spacing, is a type of letter spacing. In a kerned font, some characters are allowed to overhang others and, therefore, occupy an area that is less than each character's increment for that font. Usually this feature is restricted to pairs of characters whose appearance might benefit from greater proximity. The five character pairs that most commonly benefit from kerning are *Yo*, *We*, *To Tr*, and *Ta*.

Kerning is not available in the system-provided image fonts. The Helvetica and Times New Roman outline fonts provided with the system do include kerning information. You can specify kerning as a requirement on the GpiCreateLogFont. Selecting a kerned font lets kerning take place whenever it is defined.

The *a-space* is the area of the character cell before the character; the *b-space* is the area of the character cell occupied by the character; the *c-space* is the area of the character cell after the character. The a-space and c-space are shown in the following figure.



a-Space, b-Space, and c-Space

The best way to implement kerning is to use a kerning-pair table. The kerning table will exist only for certain pairs of characters for which adjustments are desirable.

Each of the entries in a kerning-pair table contains the code points of a pair of kerned characters and an adjustment that should be applied to the character width of the first character of the pair. A negative value means that the space is decreased, and a positive value means it is increased, whenever the two characters in a kerning pair are produced together. A kerning-pair table might be provided with a purchased font. If so, it will be recorded in the font header.

If kerning is specified on a character-pair basis (that is, if the font has a kerning-pair table), the font has a specific number of kerned pairs. GpiQueryKerningPairs returns kerning-pair information from the current logical font. For each pair of kerned characters, you are given the characters themselves and an adjustment in world coordinates that must be applied to the character width of the first character of the pair.

When you specify negative a- and c-space values, the values affect *any* character paired with the kerned character. Because this form of kerning cannot be applied selectively, you have to choose the kerned characters carefully. For example, an italic letter f (*f*) is a good candidate for this sort of kerning, because it can overhang most of the characters that precede and follow it.

An application can adjust the amount of space between all characters in a font with GpiSetCharExtra. Any extra space is added to font kerning values. An application also can adjust the width of a font's break character (space) with GpiSetCharBreakExtra.

Kerning is not available on all devices. On devices that support kerning, kerning is enabled by default. When kerning is not supported on a device, kerning support is switched off by default. When kerning support is switched off, the kerning information supplied with a font is ignored. To determine whether a device supports kerning, use DevQueryCaps.

The kerning information can be implemented by your application when a character string primitive is written to an output device. This is called *rendering* the text. GpiCharStringPos and GpiCharStringPosAt, permit specification of the starting position for each character. The entire character string can be searched for character pairs that also are kerning pairs for the kerning information to be applied.

FATTRS Data Structure

The FATTRS structure is used to identify the characteristics of a requested font. Certain attributes of a font that govern many individual

character string primitives are contained in FATTRS, as follows:

- Record length
- Selection indicators
- Match value
- Face name
- Registry identifier
- Code page
- Maximum baseline extent
- Average character width
- Type indicator
- Font use indicator

Record Length

Record length determines the length of the FATTRS structure. An application normally specifies this with the **sizeof** operator.

Selection Indicators

There are five selection indicators:

Indicator	If set ON, selects...
FATTR_SEL_ITALIC	A font with an italic appearance. This is intended for use with an image font and results in a simulation of an italic font.
FATTR_SEL_BOLD	A boldface version of the font. This is intended for use with an image font and results in a simulation of a boldface font.
FATTR_SEL_OUTLINE	Hollow characters when using an outline font. This only can be used with outline fonts. This indicator is less important than the others because improved performance is now available (from Adobe Type Manager) on filled outline fonts than was originally available when this option (FATTR_SEL_OUTLINE) was introduced.
FATTR_SEL_STRIKEOUT	A font in which the characters are struck through. This option also might be selected on the character string primitive.
FATTR_SEL_UNDERSCORE	A font in which the characters are underscored. This option also might be selected on the character string primitive.

Any combination of these indicators can be set. When set, they cause a simulation of the selected feature. For example, if you require a bold, italicized version of a PM image font, PM simulates that feature because bold, italic image fonts are not supplied.

Match Value

The match value is a number that identifies a physical font. It is allocated to the font when the font is loaded. If the match value is less than 0, the font is a *device font*; and, if the match value is greater than 0, the font is a *generic font*.

A device font is exclusive to the device with which it is associated, and its match number is guaranteed to be unique only within the device driver. If an application prints on a different printer, the application cannot guarantee that the font will be identical just by specifying the match value. A generic font is available on more than one device. The PM-supplied fonts, for example, are generic fonts. Printer fonts always are device fonts, but you also can and should use generic fonts with printers.

You can choose whether or not to use the match value to specify the font when calling GpiCreateLogFont. Specifying the match value is easier but less flexible than the alternative, which is to specify the font name and outline characteristics. For an image font, the alternative is to specify the maximum baseline extent and average character width of the required font.

Face Name

The face name value in the FATTRS data structure is identical to the face name of the FONTMETRICS data structure when describing a specific font.

Registry Identifier

The registry identifier is a unique number by which the font is registered with IBM. The registry identifier of a particular font can be retrieved by calling GpiQueryFonts. If you do not have a registry number, set this value to 0.

Code Page

The code page is the identifier of the code page to be associated with the font. You can default the code page in GpiCreateLogFont by specifying 0. Like the face name, the code page in the FATTRS data structure is identical to the code page in the FONTMETRICS data structure when describing a specific font.

Maximum Baseline Extent

The maximum baseline extent is the vertical space occupied by characters in the font. If you are setting the font-use indicator FATTR_FONTUSE_OUTLINE, you should set the maximum baseline extent to 0. Outline fonts take an equivalent value from the character cell attribute that is current when text is written to an output device.

The maximum baseline extent value is required to select an image font and must be specified in world coordinates. For image fonts, this is the vertical height in world coordinates of character images in the font. This field must be specified when requesting an image font with GpiCreateLogFont.

The maximum baseline extent measurement is shown in a previous figure. The maximum baseline extent in the FATTRS data structure is used for programming, unlike the maximum baseline extent in the FONTMETRICS data structure, which is only measurement as recommended by the font's designer.

Average Character Width

The average character width of a font is the average character increment in world coordinates. The character increment is the sum of the a-space, b-space, and c-space of a character. If you are setting the font-use indicator FATTR_FONTUSE_OUTLINE, you can set the average character width to 0 because the outline fonts take an equivalent value from the character cell attribute that is current when text is written to an output device.

This field should be specified when requesting an image font using GpiCreateLogFont. It must be specified in world coordinates.

Type Indicator

PM provides a type indicator to allow the further specification of font most appropriate to your application. The type indicators found in the FATTR data structure are not interchangeable with the type indicators in the FONTMETRICS data structure.

The most commonly used type indicator, FATTR_TYPE_KERNING, causes kerning to take effect for those fonts that contain kerning-pair information. FATTR_TYPE_KERNING is supported only for PostScript** devices.

There are three other type indicators provided:

Indicator	When...
FATTR_TYPE_ANTIALIASED	An anti-aliased font is required
FATTR_TYPE_MBCS	A mixed single- or double-byte code page is required
FATTR_TYPE_DBCS	A double-byte code page is required.

Font-Use Indicators

The font-use indicators tell the PM what you intend to do with the font, and determine whether you get an image font or an outline font. There are three font-use indicators:

Indicator	If set ON...
FATTR_FONTUSE_OUTLINE	Forces selection of an outline font. This value must be set if you intend to use graphics characters in a path definition.
FATTR_FONTUSE_TRANSFORMABLE	Allows the scaling, rotating, or shearing of font characters. Set this value ON if the current character mode is CM_MODE3.
FATTR_FONTUSE_NOMIX	Allows the PM to ignore current mix-mode values. Set this value ON if the graphics text is not going to be written over or underneath other graphics, because it improves the performance of text drawing.

FATTR_FONTUSE_TRANSFORMABLE and FATTR_FONTUSE_OUTLINE normally are both specified to select an outline font. Specify neither if you want to select an image font.

Public and Private Fonts

The fonts supplied by PM are loaded automatically when the system is started and are available to any application at any time. They are said to be *public fonts* . Any font that you define locally has to be loaded before you can use it. If you load a locally defined font using the **Install** option of the Workplace Font Palette, that font is available to any application in the system, and, therefore is a public font. Some devices provide their own fonts. All device fonts are public fonts.

Any font loaded using GpiLoadFonts from within an application is a *private font*. A private font is available only to the application that loads it. The file in which you store a locally created font definition can contain more than one font definition. GpiLoadFonts loads a named font file and, therefore, loads all fonts defined in that file. GpiQueryFontFileDescriptions provides the family name and the face name of each font in a locally defined font file. Use this function to determine if a particular file contains the font you want to use before you load it.

To unload a previously loaded private font file, call GpiUnloadFonts. You cannot unload public fonts with this function.

You cannot use private fonts in a document that you want to print using the spooler. If you want to print the document from a printer server, using the PM_Q_STD format, all the fonts used in the document must be installed as public fonts on the server. You can use either public or private fonts if you specify the data format PM_Q_RAW or if you want to print without the spooler.

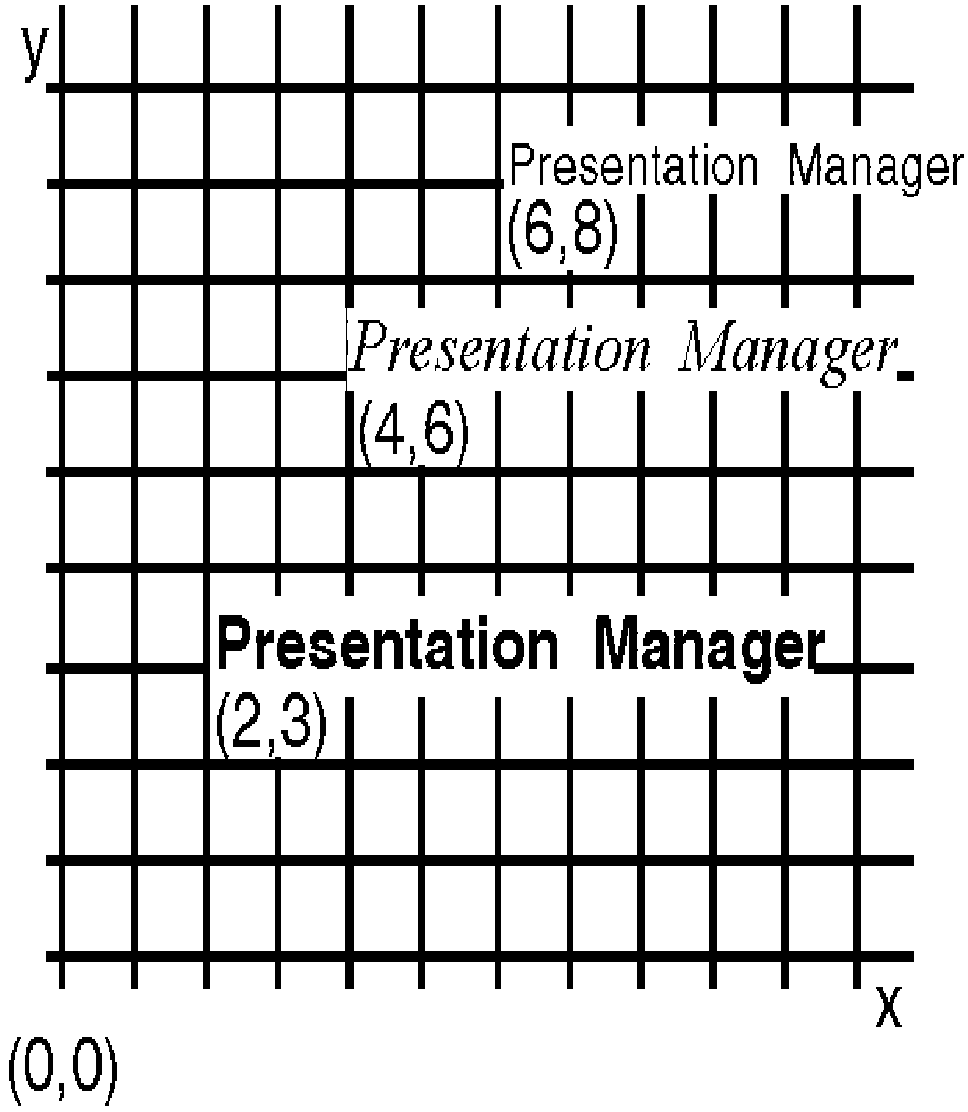
Drawing a Character String Primitive

To draw a string of one or more graphics characters from the current font, call GpiCharString. The string starts at the current position and, when it has been drawn, the new current position is the point at which the next character would be, had there been one.

To draw the string starting at a position other than the current position, call GpiCharStringAt. This is equivalent to calling GpiSetCurrentPosition or GpiMove, followed by GpiCharString. You supply the text of the character string as a parameter to the appropriate GPI function.

Note: You also can call WinDrawText, which draws text to the screen a line at a time. WinDrawText differs from the GpiCharString... functions in that WinDrawText ignores the DM_RETAIN drawing mode and can only be used to draw to a window device context.

The following figure shows some examples of a character string primitive.



The Character String Primitive

This example shows the string *Presentation Manager* drawn from different logical fonts.

Font Files and Dynamic-Link Libraries

You can use the Font Editor to alter and customize image-font files. Files created with the Font Editor have a .FNT extension. After you create a custom font, put it into a dynamic-link library (DLL) that the application can load. Once the application loads this library, it can use any of the custom fonts the library contains.

DLLs that contain fonts are unique; they contain data segments and empty (or useless) code segments. They are unique because they contain no executable code, unlike normal DLLs. Also you are able to install the font file as a public font if desired; GpiLoadFonts will make it available only as a private font. The operating system naming convention for a DLL that contains font information end with a .FON extension.

To create a DLL, you must use an assembler, a linker, and the operating system Resource Compiler. For this example, assume that your custom font file is named NEWFONT.FNT.

After creating your custom font file, you need to create a special assembler language file with your editor. Call this file MYFONT.ASM, for

example. The following figure shows an example of the source code for this file.

```
code segment word      ;Makes dummy code segment aligned on word boundary
db "empty_segment"     ;Initializes a string in dummy segment
code ends              ;Dummy segment ends here
end                    ;Terminates source file
```

After you have created MYFONT.ASM, assemble it by using the following command:

```
masm myfont
```

After assembling the MYFONT.ASM file, create a module-definition file. Call this file MYFONT.DEF, for example. It should contain the statements in the following figure.

```
LIBRARY myfont
SEGMENTS CODE MOVEABLE
```

The first statement tells the linker that you are creating a library with the module name, MYFONT. The second statement tells the linker that the segments in this library are movable code segments.

After creating MYFONT.DEF, start the linker with the command in the following figure.

```
link myfont,,,,myfont.def
```

This command creates an empty executable file called MYFONT.EXE.

After creating the empty executable file, which is the template for a DLL, create a resource file and call it MYFONT.RC. For example, if your font file is called NEWFONT.FNT, you would place the statement in the following figure in MYFONT.RC.

```
FONT    200     NEWFONT.FNT
```

This statement assigns the identifier, 200, to the font resource NEWFONT.FNT.

Finally, use the Resource Compiler to add the font file (NEWFONT.FNT) to the empty DLL as in the following figure.

```
rc  myfont.rc
```

The executable file, MYFONT.EXE, now contains your custom fonts. After you have renamed this file, MYFONT.FON, you can load the fonts into your application using GpiLoadFonts and pass it a pointer to the path and library name as the second argument-for example, c:\os2\dll\myfont.fon.

Using Fonts

You can use the font functions to:

- Determine the characteristics of available fonts
- Select a font for text output
- Delete or unload a font when finished
- Create a font
- Create marker sets and pattern sets
- Use a bit map as a fill pattern
- Use a clip path as a fill pattern

Selecting a Font

Every presentation space has a *current font*, and graphics characters must be drawn in that font. By default, the current font is the system font. Before an application can use any font other than the system font, it has to identify the other font as the new current font.

There are two distinct ways to identify the other font:

- Call GpiQueryFonts to request information about the available physical fonts, and then explicitly select one of those fonts by supplying its identifier as input to GpiCreateLogFont.

 This process has 11 steps and is covered in the topic "Explicit Font Selection." This method of selecting a font is most useful for selecting image fonts.

 If you must have an image font, you are urged to use this "explicit selection"method, because you need to specify a maximum baseline extent and an average character width in the FATTRS structure. These values can be determined only by calling GpiQueryFonts. If you specify values that are not valid, you are likely to be given an outline font.

- Use GpiCreateLogFont to specify the features you require in a font, and permit the PM programming interface to make an appropriate font available to you.

 This process has two steps and is covered in the topic "Closest Matching Font Selection."

Explicit Font Selection

An application can select either a public or a private font with GpiCreateLogFont. A public font is available to all applications. A private font is loaded by an application for its exclusive use.

Use the Presentation Manager Control Panel to load a public font. Four DLLs contain the Times Roman, Helvetica, Courier, and System Monospaced image fonts. The names of these libraries are:

- TIMES.FON
- HELV.FON
- COURIER.FON
- SYSMONO.FON

Unlike most DLLs, font libraries typically use the file name extension .FON. If the user loads all four libraries, a total of 76 public fonts are available. An application can use both outline and image formats of these fonts. Characters in the image format are available in sizes ranging from 8 to 24 points. Characters in the outline format can be any size.

Call GpiLoadFonts to load a private font. Pass the function the path and name of the DLL that contains the font. After the application loads the DLL of fonts, it can determine the characteristics of the fonts in that library by calling GpiQueryFonts.

To select a public font from all of the available public fonts, do the following:

1. Call GpiQueryFonts. passing the QF_PUBLIC flag, a NULL pointer to the font's face name, and a count of 0 to determine the number of available public fonts, as shown in the following figure.

```
#define INCL_GPILCIDS
#include <os2.h>
void fncFONT06(void){

    FONTMETRICS fm, afm[80];
    LONG cFonts = 0, cPublicFonts;
    HPS hps;

    cPublicFonts = GpiQueryFonts(hps,      /* Queries all public fonts */
    QF_PUBLIC,
    (PSZ) NULL,
    &cFonts,
    sizeof(fm),
    (PFONTMETRICS) NULL);
} /* fncFONT06 */
```

 Note: To load and use a private font, follow the same procedure, but specify QF_PRIVATE, instead of QF_PUBLIC, in the calls to GpiQueryFonts.

2. Copy the value returned by GpiQueryFonts into a LONG integer variable.

3. Allocate memory for the FONTMETRICS data structures.

4. Call GpiQueryFonts again, passing the QF_PUBLIC flag, a NULL pointer to the font's face name, the count returned by the previous call, and the address of an array of FONTMETRICS structures.

 GpiQueryFonts the font metrics of every font available the array of FONTMETRICS structures. The font metrics define in detail the physical characteristics of a font.

 Because the font metrics are so detailed, the amount of information returned to you from GpiQueryFonts can be extensive. You can restrict the amount of information returned by this function by:

 - Specifying an absolute number of fonts about which you require information.
 - Asking for information about fonts of a specific face name only.
 - Limiting the length of the font-metrics buffer.

 If you request information for more fonts than are available on the system, GpiQueryFonts returns all the information about the available fonts and a value indicating how many fonts it has returned.

 If you request information for fewer fonts than are available on the system (that match the specified face name, and so forth), GpiQueryFonts returns a value indicating the number of fonts that it was unable to return.

 The following figure is an example of this technique of selecting a font.

```
#define INCL_GPILCIDS
#include <os2.h>

PFONTMETRICS fncFONT07 (HPS hps, PLONG pcPublicFonts) {
   PFONTMETRICS afm = NULL;
   LONG         fonts = 1000;

   *pcPublicFonts = GpiQueryFonts(hps,
                                  QF_PUBLIC,
                                  (PSZ) NULL,
                                  &fonts,
                                  sizeof(*afm),
                                  (PFONTMETRICS) NULL);

   if (!DosAllocMem(&afm,
                    sizeof(*afm)*(*pcPublicFonts),
                    PAG_COMMIT | PAGWRITE )) {
       GpiQueryFonts(hps,
                    QF_PUBLIC,
                    (PSZ) NULL,
                    pcPublicFonts,
                    sizeof(*afm),
                    afm);
   } /* endif */
   return afm;
}
```

5. Examine the array of FONTMETRICS structures.

 a. For an outline font, examine the face name and other attributes of the font your application requires.

 b. For an image font, examine the device resolution fields. These fields are the ones that determine if a particular font and its metrics match the device with which you wish to use it.

 Note: If your application is interactive, organize the face names and other information relevant to those fonts (such as point sizes) in a menu. When the operator has selected a font, supply the relevant information from the font metrics of that font in the FATTRS structure of GpiCreateLogFont.

6. After determining which font to use, your application must fill in the fields of the FATTRS structure. Some of the data for these fields will be copied (explicitly) from the FONTMETRICS structure. Then call GpiCreateLogFont.

 If you do not supply a face name, the default font is used. If you supply a face name, and the presentation driver cannot find a matching font, a default font is selected.

 The FATTRS structure describes a *logical font*. An application can have up to 254 logical fonts defined at any one time in a single presentation space. A logical font is a list of font attributes, whereas a *physical font* is the bit-map or vector information that the system uses to draw characters.

7. The fonts supplied by the PM programming interface, the fonts you load using the Control Panel, and any fonts you load with GpiLoadFonts are all *physical fonts* .

 Copying the entries in the FATTRS structure ensures that, if a particular font is unavailable, an attempt is made to find the most suitable substitute. Without the FATTRS information, PM is less likely to locate a suitable font.

8. Initialize a local identifier (lcid) for the new font.

9. Call GpiCreateLogFont and pass as input parameters:

 - The lcid for the font
 - The address of an 8-character array containing the name you want to give to the logical font, (or, if you do not want to specify a name, the value NULL)
 - The address of the FATTRS structure

10. Examine the return value from GpiCreateLogFont. The value will be **2** if the function is successful; **1**, if the default font was used.

11. Pass the lcid to GpiSetCharSet, assigning the font to your application's presentation space. Then the application can use it for text output. (This step identifies a logical font as the current font.)

The following figure shows how to select an image font at least 14 device coordinate units high.

```
#define INCL_GPILCIDS
#include <os2.h>

LONG fncFONT08(HPS hps, LONG lcid, LONG xres, LONG yres) {
   FATTRS       fat;
   PFONTMETRICS afm;
   LONG         cFonts;
   LONG         i;
   LONG         rc = 0;

   afm = fncFONT07(hps, &cfonts);
   if (amf) {
      for (i=0; 1<cfonts; i++) {
         if (!(afm[i].fsDefn & FM_DEFN_OUTLINE)
            afm[i].sXDeviceRes == xres &&
            afm[i].sYDeviceRes == yres &&
            afm[i].lMaxBaselineExt >= 14)  {
           fat.usRecordLength = sizeof(fat);
           fat.fsSelection = 0;
           fat.lMatch = 0;
           strcpy(fat.szFacename, afm[i].szFacename);
           fat.idRegistry = afm[i].idRegistry;
           fat.usCodePage = 0;
           fat.lMaxBaselineExt = afm[i].lMaxBaselineExt;
           fat.lAveCharWidth = afm[i].lAveCharWidth;
           fat.fsType = 0;
           fat.fsFontUse = 0;

           rc = GpiCreateLogFont(hps, (PSTR8) NULL, lcid, &fat);
           if (rc) {
              GpiSetCharSet(hps, lcid);
           } /* endif */
        } /* endfor */

     } /* endif */

     return rc;
}
```

GpiQueryFonts returns device coordinates for image fonts. For outline fonts, it returns *notional coordinates* . Notional coordinates are the coordinate in which the font was defined. Usually outline fonts are defined over a 1000-by-1000 matrix, with the unit of the matrix a 1/1000 of the em height.

Reassociating the Presentation Space

The match value of any device font is guaranteed to be unique only for the current device. If you associate the presentation space with a different device context, any logical fonts that also are device fonts are no longer available. On the new device, the match number could identify a different device font (which may not be suitable) or no font at all. Therefore, you should redefine a logical font and select it as the current font if you reassociate the presentation space when the current font is a device font. Otherwise, the default font is used.

If you do not intend to use the font on the new device, however, you need not redefine it. You will be able to continue using the font if you reassociate the presentation space with the original device context.

Any generic fonts can continue to be used after the presentation space is reassociated if they are suitable for the new device. An image font, for example, can be used only on devices that are all-points addressable.

Font Resolution

Device fonts, particularly printer fonts, usually are designed for a specific resolution, namely the resolution of the device on which they are to be used. By contrast, generic outline fonts are not device-specific. Generic image fonts are designed for a particular resolution value and, therefore, are really applicable only to devices with a matching resolution value.

The horizontal and vertical font resolution values for the device are returned by DevQueryCaps. The desired horizontal and vertical device resolution values in the font metrics of the font concerned are returned by GpiQueryFonts. Compare these the device and the desired resolution values only for an image font. If they are identical then this image font is designed for use with this device (for example, the point size in the font metrics will be accurate for this device). To select a font, do the following:

1. Query the available fonts using GpiQueryFonts.
2. If you want an outline font, search the metrics of fonts returned by GpiQueryFonts for a font that is an outline font and that has the required face name.
3. If you want an image font, search the metrics of the fonts returned by GpiQueryFonts for a font that is an image font and that has the following:
 - Required face name
 - Required point size
 - Horizontal device resolution font metric that matches the horizontal resolution for the device
 - Vertical device resolution font metric that matches the vertical resolution for the device.
4. As an alternative to the above for an image font, search the metrics of the fonts returned by GpiQueryFonts for a font that is an image font and that has the following:
 - Required face name
 - Required em height in world coordinates.

Before you define a logical font, determine the resolution of the current device by using DevQueryCaps with the CAPS_VERTICAL_FONT_RES and CAPS_HORIZONTAL_FONT_RES parameters.

Compare the values returned from this function with the **xDeviceRes** and **yDeviceRes** values in the font metrics of the available fonts. These two values contain the resolution of the device for which the font was designed. All measurements are in pels per inch. By comparing the font resolution with the device resolution, you can identify the image font best suited to the current device.

Closest-Matching Font Selection

The alternative to selecting a specific font is to set the match value to 0. To have the Presentation Manager programming interface select the closest-matching font available, do the following:

1. Define a logical font with GpiCreateLogFont. This function establishes a link between the calling application and an appropriate physical font. GpiCreateLogFont accepts a number of options as input, including the *font attributes* , which describe the physical features and capabilities of the font.

2. Complete the FATTRS structure of GpiCreateLogFont. The FATTRS structure describes a logical font, which the system uses to find the closest matching physical font. Depending on whether you are selecting a font using a match number, fill in the FATTRS structure, observing the rules listed in the following table.

Filling in the FATTRS Structure

If using a match number	If not using a match number
Set the match number to the required FONTMETRICS value. The FONTMETRICS value is the value in the FONTMETRICS structure for the required font returned by GpiQueryFonts.	Set the match number to zero.
Set the maximum baseline extent to zero.	If your font is an image font: Set the maximum baseline extent to the required FONTMETRICS value.
Set the average character width to zero.	Set the average character width to the required FONTMETRICS value.
Note: A negative lMatch is only unique for a device. A positive lMatch is only unique on a particular system at a particular time.	

Also observe the rules concerning the font-use indicator that are listed in the following table.

Font-Use Indicator Considerations

If you have *not* set the FATTR_FONTUSE_OUTLINE indicator:	If you have set the FATTR_FONTUSE_OUTLINE indicator:
PM looks for an image font that has the required selection indicators and whose maximum baseline extent and average character width match those you have specified.	PM looks for a suitable outline font whose selection indicators match those you have specified.
If no suitable image font is found, PM looks for an outline font with the required face name and selection indicators.	If no suitable outline font is found, the default font is used. No attempt is made to substitute an image font.
If no suitable outline font is found, the default font is made available to you.	

Outline fonts are affected by the current character attributes (for example, character box, shear, and angle). Because an outline font might be made available to you, even when you requested an image font, call GpiQueryFontMetrics to determine whether the font is an image font or an outline font. GpiQueryFontMetrics returns the metrics of the current logical font in world coordinates.

The setting of the definition indicators in the FONTMETRICS structure tell you whether the font is either an image or outline font, or a device or generic font. If the font is an outline font, specify values for the character attributes before using the font. Outline fonts are unaffected by the maximum baseline extent and average character width values specified in GpiCreateLogFont.

If the default font provided is an image font, and you also have specified FATTR_FONTUSE_TRANSFORMABLE, any attempt to draw graphics characters in CM_MODE3 raises an error condition. This is an exception to the general rule that a FATTR_FONTUSE_TRANSFORMABLE font must be used in CM_MODE3 only.

Selecting the New Current Font

In addition to completing the FATTRS structure, you must supply as input to GpiCreateLogFont a unique lcid for the font. The identifier is a number in the range of 1 through 254. Because you can have more than one logical font definition in a presentation space, you must select the current font by supplying the font's lcid as input to GpiSetCharSet.

You do not have to call GpiSetCharSet to use the default font. When you want to reset the current font to its default value, however, call GpiSetCharSet with the value LCID_DEFAULT. The current font, like other modal attributes, reverts automatically to its default value when specific GPI functions are called.

Deleting Logical Fonts and Unloading Physical Fonts

A logical font can be used only by the application that defines it. The logical font definition is stored in the presentation space and is saved on the LIFO stack with its lcid when GpiSavePS is called. Unlike a physical font definition, a logical font definition is temporary and is lost when the presentation space is deleted.

To specifically delete a logical font, call GpiDeleteSetId, which deletes the logical font, and makes its lcid available for reuse. Sometimes this is necessary because there are only 255 lcids available. The logical font must not be the current font when you call GpiDeleteSetId.

Logical fonts are deleted to release the lcid and free memory. Unload private fonts using GpiUnloadFonts to free memory and to make them no longer available. Uninstall public fonts using the Control Panel to free memory and to make them no longer available.

You must end the association between physical and logical fonts *before* calling GpiUnloadFonts.

If you call GpiUnloadFonts to unload a private font file, you must end the association between the physical fonts in that file and the logical fonts defined by the application. Do the following:

- Call GpiSetCharSet to select a current font (such as the default font) that is not linked to the physical fonts you are unloading
- Call GpiDeleteSetId to delete each logical font that refers to any of the physical fonts in the font resource file. If you fail to do this, an error is returned to your application from GpiUnloadFonts. *GpiUnloadFonts.

Creating Fonts

You cannot edit the definitions of any of the PM-provided fonts, although you can create and edit image-font files using the Font Editor. A font file contains a header, that describes the font in general terms and a section containing bit maps of the characters themselves. The font definition is stored in a resource file.

Creating Marker Sets and Pattern Sets

Like character sets, marker sets and pattern sets are defined in fonts. For both marker sets and pattern sets, there are system-provided defaults. You also can create your own image markers or patterns, but not outline markers or patterns, using the Font Editor. The font definition containing the image or pattern can be loaded either as a public font (using the **Install** selection from the Control Panel) or as a private font (using GpiLoadFonts). The current character mode has no effect on marker sets and pattern sets. Any character from a font can be used as a marker or as an area-fill pattern.

Before you can use a locally defined marker set or pattern set, you must call GpiCreateLogFont to define a logical font and to give it an lcid, just as you would for a character font. Most of the values you can specify with GpiCreateLogFont are not applicable to marker sets and

pattern sets. Furthermore, in the instance of marker sets and pattern sets, you usually know that you want to use a particular font.

To select a locally defined marker set or pattern set:

1. Call GpiQueryFonts to obtain a match value for the font.
2. Call GpiCreateLogFont, specifying the face name, match value, and lcid for the font.
3. Select the logical font as the current marker set or pattern set by calling GpiSetMarkerSet or GpiSetPatternSet, as appropriate.

To revert to using the default pattern or marker sets, call GpiSetPatternSet or GpiSetMarkerSet with the value LCID_DEFAULT.

Using Bit Maps as Area-Fill Patterns

An alternative to using the Font Editor to create your own pattern sets is to use a bit map as an area-fill pattern. You must call GpiSetBitmapId to give the bit map an lcid, and then call GpiSetPatternSet, specifying the lcid of the bit map. The bit map becomes a pattern set containing one pattern, and the current pattern-symbol attribute is ignored. GpiQueryBitmapHandle returns the handle of the bit map currently known by the lcid that you supply as input.

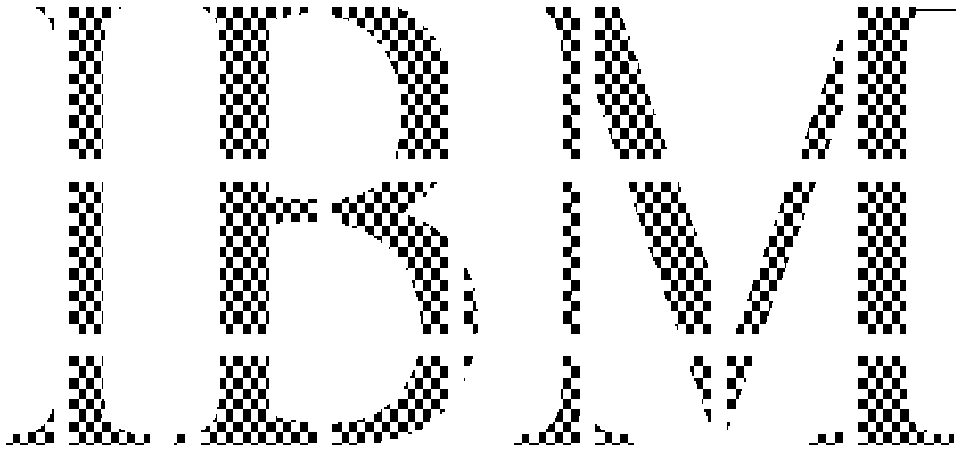
PM cannot distinguish among the lcids associated with a particular presentation space. Any lcid can identify a character set, marker set, pattern set, or bit map (that is being used as an area-fill pattern). For example, if you call GpiQueryNumberSetIds to determine how many lcids are currently assigned in the presentation space, the number returned to you includes all logical fonts defined for this presentation space. You also can use GpiDeleteSetId to remove lcids associated with the non-font entities. If you call this function to delete a bit map that is being used as an area-fill pattern, the bit map itself is not deleted. However, it is disassociated from its lcid, and the lcid can be reused.

Using Paths with Fill Patterns

Using paths lets you fill a character string with a complicated or nonrepeating pattern. The following sequence is recommended:

- Select the required outline font, and set the required character attributes.
- Create a path by calling GpiBeginPath, GpiCharString, and GpiEndPath.
- Set the required pattern symbol and other AREABUNDLE pattern attributes.
- Fill the path using GpiFillPath.

The following figure provides an example of using paths with fill patterns to create characters.



Using Clip Paths to Create Characters

Font-File Format

The OS/2 font-file format consists of two sections. The first section contains the general attributes of the font, and describes features such as its typeface, style, and nominal size. The second section contains the actual definitions of the characters belonging to the font.

The font resource is a set of self-defining records of the form:

```
typedef struct _RECORD {
        ULONG   ulIdentity;    /* structure identity code */
        ULONG   ulSize;        /* structure size in bytes */
          .                    /* data */
          .
          .
} RECORD;
```

A font starts with a special font-signature structure and ends with an ending structure. The font signature has the form:

```
typedef struct _FONTSIGNATURE {
        ULONG   ulIdentity;
        ULONG   ulSize;
        CHAR    achSignature [12]
} FONTSIGNATURE;
```

Where:

ulIdentity	0xFFFFFFFE
ulSize	20
achSignature	"OS/2 FONT" for an OS/2 1.x format font, or "OS/2 FONT 2" for an OS/2 2.x format font.

A 2.x format font includes additional font description information in the PANOSE structure. This structure will be added to the end of the .FNT file (prior to the ENDFONT record).

The font end structure has the form:

```
typedef struct _ENDFONT{
        ULONG   ulIdentity;
        ULONG   ulSize;
}ENDFONT
```

Where:

ulIdentity	0xFFFFFFFF
ulSize	8

All records should be in the order of their identity fields.

There are three or four records in a font resource between the font signature and the font end:

- The font metrics
- The font character definitions
- The pair kerning table.
- The PANOSE description (for "OS/2 FONT 2" fonts).

Following compilation, the records in the resource are in the order defined above.

Metric Information Contained in Fonts

This section gives an explanation of how to set the fields of the FOCAMETRICS structure when developing:

- A bit map or outline font for general use by PM graphics applications
- A description of a bit map or outline device font that is built in to a device or can be downloaded to a device.

The following structure contains the physical font metrics used when creating fonts. It is defined in the file \INCLUDE\PMFONT.H.

```
typedef struct _FOCAMETRICS {
        ULONG   ulIdentity;
        ULONG   ulSize;
        CHAR    szFamilyname[32];
        CHAR    szFacename[32];
        SHORT   usRegistryId;
        SHORT   usCodePage;
        SHORT   yEmHeight;
        SHORT   yXHeight;
        SHORT   yMaxAscender;
        SHORT   yMaxDescender;
        SHORT   yLowerCaseAscent;
        SHORT   yLowerCaseDescent;
        SHORT   yInternalLeading;
        SHORT   yExternalLeading;
        SHORT   xAveCharWidth;
        SHORT   xMaxCharInc;
        SHORT   xEmInc;
        SHORT   yMaxBaselineExt;
        SHORT   sCharSlope;
        SHORT   sInlineDir;
        SHORT   sCharRot;
        USHORT  usWeightClass;
        USHORT  usWidthClass;
        SHORT   xDeviceRes;
        SHORT   yDeviceRes;
        SHORT   usFirstChar;
        SHORT   usLastChar;
        SHORT   usDefaultChar;
        SHORT   usBreakChar;
        SHORT   usNominalPointSize;
        SHORT   usMinimumPointSize;
        SHORT   usMaximumPointSize;
        SHORT   fsTypeFlags;
        SHORT   fsDefn;
        SHORT   fsSelectionFlags;
        SHORT   fsCapabilities;
        SHORT   ySubscriptXSize;
        SHORT   ySubscriptYSize;
        SHORT   ySubscriptXOffset;
        SHORT   ySubscriptYOffset;
        SHORT   ySuperscriptXSize;
        SHORT   ySuperscriptYSize;
        SHORT   ySuperscriptXOffset;
        SHORT   ySuperscriptYOffset;
        SHORT   yUnderscoreSize;
        SHORT   yUnderscorePosition;
        SHORT   yStrikeoutSize;
        SHORT   yStrikeoutPosition;
        SHORT   usKerningPairs;
        SHORT   sFamilyClass;
        PSZ     pszDeviceNameOffset;
} FOCAMETRICS;
```

Note: FOCAMETRICS is a parallel structure with FONTMETRICS as returned to applications in the GpiQueryFonts and GpiQueryFontMetrics function calls.

The FONTMETRICS fields are derived from FOCAMETRICS by the Presentation Manager graphics engine. Most values are passed though unchanged. The exceptions are:

- The *Identity* field. This must be 1. This field is not a part of the FONTMETRICS structure.
- The *Size* field. This must be set to the size of the FOCAMETRICS structure. This field is not a part of the FONTMETRICS structure.
- The *Codepage* field. Ignore the description in FONTMETRICS, and use the following:

 Place 850 in this field if the font is intended to support any PM supported code page.

 Place 65400 in this field if the font has special glyphs, for example if it is a Symbol font.

Place other valid code pages in this field if the font is specific to this code page.

Do not place other values in this field.

- FONTMETRICS fields which contain values in world coordinates. The corresponding field in FOCAMETRICS should contain pel values for bit-map fonts, and notional units for outline fonts.

See FONTMETRICS for a detailed explanation of the fields.

Font Character Definitions

Two formats of font character definition are supported. These are:

Image format The character glyphs are represented as pel images.

Outline format The character glyphs are represented by vector data that traces the outline of the character.

Note: Intelligent Font Technology fonts (such as ATM Type-1 fonts) may be stored in a technology specific format, and thus will not conform to this definition for outline fonts.

The definition consists of a header portion and a portion carrying the characters themselves.

The header portion contains information about the format of the character definitions and data about each character including width data and the offset into the definition section at which the character definition begins. (See Definitions of Terms Used When Describing Fonts.)

1. Proportional characters (a+b+c = character increment) for each character:
 a,b,c >= 0

2. Characters where a, b, and c are definitions for all characters:
 b >= 0
 a, c any integer

Raster fonts contain a "null character". The character definition record for this occurs after the one for the last character. Thus the format has usLastChar+2 characters, although the null character is not counted in the range returned. The null character is composed of zeros and is always eight pels wide.

Font Definition Header

This structure defines the format or the character definition records that follow it:

```
typedef struct_FONTDEFINITIONHEADER {
        ULONG        ulIdentity;
        ULONG        ulSize;
        SHORT        fsFontdef;
        SHORT        fsChardef;
        SHORT        usCellSize;
        SHORT        xCellWidth;
        SHORT        yCellHeight;
        SHORT        xCellIncrement;
        SHORT        xCellA;
        SHORT        xCellB;
        SHORT        xCellC;
        SHORT        pCellBaseOffset;
} FONTDEFINITIONHEADER;
typedef FONTDEFINITIONHEADER FAR *PFONTDEFINITIONHEADER;
```

Where:

ulIdentity
4 bytes.

Must be equal to 2.

ulSize

4 bytes.

Size of this structure in bytes.

fsFontdef

2 bytes of flags.

Indicates which fields are present in the font definition data in the header.

Type 1

Bit 0	1 = width defined in header
Bit 1	1 = height defined in header
Bit 2	1 = char increment same as width, so that it is defined for the whole font
Bit 3	0 = a-space not defined
Bit 4	0 = b-space not defined
Bit 5	0 = c-space not defined
Bit 6	1 = base offset same for all characters.

Type 2

Bit 0	0 = width for each character unique
Bit 1	1 = height defined in header
Bit 2	0 = char increment same as width, so that it is unique for each character
Bit 3	0 = a-space not defined
Bit 4	0 = b-space not defined
Bit 5	0 = c-space not defined
Bit 6	1 = base offset same for all characters.

Type 3

Bit 0	0 = width for each character unique
Bit 1	1 = height defined in header
Bit 2	0 = char increment same as width, so that it is unique
Bit 3	0 = a-space not defined
Bit 4	0 = b-space not defined
Bit 5	0 = c-space not defined
Bit 6	1 = base offset same for all characters.

FsChardef

2 bytes of flags.

Indicates which fields are present on a per character basis.

Type 1

Bit 0	1 = width defined for each character (performance op)
Bit 1	0 = height is in header
Bit 2	0 = char increment is in header
Bit 3	0 = a-space not defined
Bit 4	0 = b-space not defined
Bit 5	0 = c-space not defined
Bit 6	0 = base offset defined in header
Bit 7	1 = offset to glyph defined.

Type 2

Bit 0	1 = width defined for each character
Bit 1	0 = height is in header
Bit 2	0 = char increment same as width
Bit 3	0 = a-space not defined
Bit 4	0 = b-space not defined
Bit 5	0 = c-space not defined
Bit 6	0 = base offset defined in header
Bit 7	1 = offset to glyph defined.

Type 3

Bit 0	1 = width not defined, use a, b, c

Bit 1	0 = height is in header
Bit 2	0 = char increment same as width
Bit 3	1 = a-space defined
Bit 4	1 = b-space defined
Bit 5	1 = c-space defined
Bit 6	0 = base offset defined in header
Bit 7	1 = offset to glyph defined.

usCellSize

2-byte integer.

Indicates the length in bytes of each character definition record (the per character data).

Type 1	6 bytes
Type 2	6 bytes
Type 3	10 bytes.

xCellWidth

2-byte integer

The width of the characters, in pels for image fonts, and relative units for outline fonts.

Type 1	Width of the characters
Type 2	Zero
Type 3	Zero.

yCellHeight

2-byte integer.

The height of the characters, in pels for image fonts, and relative units for outline fonts.

Type 1	Height of the characters
Type 2	Height of the characters
Type 3	Height of the characters.

xCellIncrement

2-byte integer.

The distance along the character baseline required to step from one character to the next (when forming a character string).

Type 1	Width of the characters
Type 2	Zero
Type 3	Zero.

xCellA

2-byte signed integer.

The width of the space before a character in the inline direction (the a-space).

Type 1	Zero
Type 2	Zero
Type 3	a-space for all characters.

xCellB

2-byte integer.

The width of a character (inline direction). The b-space.

Type 1	Zero
Type 2	Zero
Type 3	b-space for all characters.

xCellC

2-byte signed integer.

The width of the space after a character in the inline direction (the c-space).

Type 1	Zero
Type 2	Zero
Type 3	c-space for all characters.

pCellBaseOffset

2-byte signed integer.

The position of the top of a character definition relative to the baseline in the direction perpendicular to the baseline.

Type 1	Baseline offset for all characters
Type 2	Baseline offset for all characters
Type 3	Baseline offset for all characters.

Character Definition Record
xCellSize bytes per record.

The following fields may or may not be present, according to the font character definition fields flags. If a field is present, it is present for *each* character and the value applies to that character only.

There are usLastChar+2 such records for raster fonts. The final one is for the null character.

- Character Definition Offset: 4-byte integer.

 The offset into the Font File at which the character definition begins.

 Data for a single character raster or vector should not span two segments; that is, if a character is too big to fit into a segment it should be put in the next segment.

 This field should be set to zero if the character being defined is a blank character.

- Character Cell Width: 2-byte integer.

 The width of the character definition in pels.

- Character Cell Height: 2-byte integer.

 The height of the character definition in pels.

- Character Increment: 2-byte integer.

 The length along the character baseline required to step from this character to the next (when forming a character string).

- Character a-space: 2-byte signed integer.

 The width of the space before the character in the inline direction.

- Character b-space: 2-byte integer.

 The width of the character shape (inline direction).

- Character c-space: 2-byte signed integer.

 The width of the space after the character in the inline direction.

- Character Baseline Offset: 2-byte signed integer.

 The position of the top of a character definition relative to the baseline in the direction perpendicular to the baseline.

Note: Type 1 fonts have offset/width pairs (like type 2); however, the usCellSize and xCellIncrement are nonzero. In the fsType field of the font metrics, the proportional-space flag, bit 0, is set.

Image Data Format

The bits for each character are stored separately, and start on a byte boundary. Sequential bytes represent vertical pieces of the character image. For example, a 15-bit-wide H is stored as follows:

```
Byte                      Byte

  1   00000000   0000000-   13      Bytes 1 through 12 are composed of
  2   01100000   0000110-   14      whole bytes of data stored row by row.
  3   01100000   0000110-   15
  4   01100000   0000110-   16      Bytes 13 through 24 are composed of
```

```
 5   01100000   0000110-   17      bytes stored row by row, where each byte
 6   01111111   1111110-   18      contains 7 bits of information and the
 7   01111111   1111110-   19      last bit is unused.
 8   01100000   0000110-   20
 9   01100000   0000110-   21      Thus the character is laid down in
10   01100000   0000110-   22      byte-wide columns.
11   01100000   0000110-   23
12   00000000   0000000-   24
```

Notes:

1. There is always an additional (null) character defined in an Image Font (defined at character position LastChar+2) which is 8 bits wide, the height of the font character, and set to all zeros.

2. The maximum size of each individual Image Font must not exceed 64KB.

The Kerning Pair Table

The kerning pair table record is not present if the _KerningPairs record in the metrics is zero. If it is present, the code points are words, not bytes. This table should be sorted by kpChar1 and kpChar2 order to allow binary searches.

```
typedef struct _KERNPAIRTABLE {
        ULONG   ulIdentity;
        ULONG   ulSize;
        CHAR    cFirstpair;
}KERNPAIRTABLE;

typedef struct _KERNPAIRS {
        SHORT   sFirstChar;
        SHORT   sSecondChar;
        SHORT   sKerningAmount;
}KERNINGPAIRS;
```

Where:

ulIdentity	3
ulSize	10
sFirstChar	First character of the kerning pair.
sSecondChar	Second character of the kerning pair.
sKerningAmount	Kerning value. Positive values increase the inter-character spacing while negative values bring the characters closer together.

Outline Data Format

Fonts defined by outlines (vectors) may contain any of these graphics orders:

- Line at given position (GLINE)
- Line at current position (GCLINE)
- Relative line at given position (GRLINE)
- Relative line at current position (GCRLINE)
- Fillet at given position (GFLT)
- Fillet at current position (GCFLT)
- Sharp fillet at given position (GSFLT)
- Sharp fillet at current position (GCSFLT)
- Bezier curve at given position (GBEZ)
- Bezier curve at current position (GCBEZ)
- No operation (GNOP1)
- Comment (GCOMT)
- End of symbol definition (GESD).

The maximum length of the data in these orders is 255 bytes. The drawing order code and the length fields are not included in the length count.

The size of each outline font definition must not be longer than 64KB.

Additional Metrics

The additional metrics structure extends the metrics describing the font to include the PANOSE fields. The fields allow for quantitative descriptions of the visual properties of font faces. The format of the ADDITIONALMETRICS structure is:

```
typedef struct {
        ULONG       ulIdentity;
        ULONG       ulSize;
        PANOSE      panose;
} ADDITIONALMETRICS;
```

Where:

ulIdentity	4
ulSize	20
panose	The ten digit PANOSE number with two bytes of padding.

The PANOSE definition consists of ten digits, each of which describes one of up to sixteen variations. The current digits are:

1. Family Kind (6 variations)

0	= Any
1	= No Fit
2	= Text and Display
3	= Script
4	= Decorative
5	= Pictorial

2. Serif Style (16 variations)

0	= Any
1	= No Fit
2	= Cove
3	= Obtuse Cove
4	= Square Cove
5	= Obtuse Square Cove
6	= Square
7	= Thin
8	= Bone
9	= Exaggerated
10	= Triangle
11	= Normal Sans
12	= Obtuse Sans
13	= Perp Sans
14	= Flared
15	= Rounded

3. Weight (12 variations)

0	= Any
1	= No Fit
2	= Very Light
3	= Light
4	= Thin
5	= Book
6	= Medium
7	= Demi
8	= Bold
9	= Heavy
10	= Black
11	= Nord

4. Proportion (10 variations)

0	= Any
1	= No Fit
2	= Old Style

3 = Modern
4 = Even Width
5 = Expanded
6 = Condensed
7 = Very Expanded
8 = Very Condensed
9 = Monospaced

5. Contrast (10 variations)

0 = Any
1 = No Fit
2 = None
3 = Very Low
4 = Low
5 = Medium Low
6 = Medium
7 = Medium High
8 = High
9 = Very High

6. Stroke Variation (9 variations)

0 = Any
1 = No Fit
2 = Gradual/Diagonal
3 = Gradual/Transitional
4 = Gradual/Vertical
5 = Gradual/Horizontal
6 = Rapid/Vertical
7 = Rapid/Horizontal
8 = Instant/Vertical

7. Arm Style (12 variations)

0 = Any
1 = No Fit
2 = Straight Arms/Horizontal
3 = Straight Arms/Wedge
4 = Straight Arms/Vertical
5 = Straight Arms/Single Serif
6 = Straight Arms/Double Serif
7 = Non-Straight Arms/Horizontal
8 = Non-Straight Arms/Wedge
9 = Non-Straight Arms/Vertical
10 = Non-Straight Arms/Single Serif
11 = Non-Straight Arms/Double Serif

8. Letterform (16 variations)

0 = Any
1 = No Fit
2 = Normal/Contact
3 = Normal/Weighted
4 = Normal/Boxed
5 = Normal/Flattened
6 = Normal/Rounded
7 = Normal/Off Center
8 = Normal/Square
9 = Oblique/Contact
10 = Oblique/Weighted
11 = Oblique/Boxed
12 = Oblique/Flattened
13 = Oblique/Rounded
14 = Oblique/Off Center
15 = Oblique/Square

9. Midline (14 variations)

0 = Any
1 = No Fit
2 = Standard/Trimmed
3 = Standard/Pointed

4	= Standard/Serifed
5	= High/Trimmed
6	= High/Pointed
7	= High/Serifed
8	= Constant/Trimmed
9	= Constant/Pointed
10	= Constant/Serifed
11	= Low/Trimmed
12	= Low/Pointed
13	= Low/Serifed

10. X-height (8 variations)

0	= Any
1	= No Fit
2	= Constant/Small
3	= Constant/Standard
4	= Constant/Large
5	= Ducking/Small
6	= Ducking/Standard
7	= Ducking/Large

When using the PANOSE number to match fonts, the ordering of the PANOSE digit is the key to finding the closest match. The most significant digit is the first digit, and the least significant digit is number ten. To find matches, the digits need to be compared, in the order given. A font mapper may want to change the precedence of the digits, to give higher weightings to other font features.

Font Directory

This section describes the directory section of a font resource. A font resource contains a directory consisting of a set of structures each containing the metrics of a font and a pointer to the font itself. This font directory is generated by the resource compiler.

The format of the font directory is:

```
typedef struct {
        USHORT      usHeaderSize;
        USHORT      usnFonts;
        USHORT      usiMETRICS;
        FONTENTRY   fntEntry[1];
} FONTDIRECTORY;

typedef struct {
        USHORT          usIndex;
        FONTFILEMETRICS metrics;
} FONTENTRY;
```

Where:

usHeaderSize	The size of the header, in bytes.
usnFonts	The number of fonts in the resource.
usiMetrics	The size of the FOCAMETRICS structures that follow the header. Note that the set of metrics for all the fonts in the resource follow the header.
usIndex	The index of a particular font; an identifier assigned to the font when the resource was created (defined in the .RC file).
metrics	The font metrics structure for the font. This is identical to a FOCAMETRICS structure with the addition of the PANOSE fields to the end.

Definitions of Terms Used When Describing Fonts

a-space, b-space, c-space - The a-space is the distance from the left of the character frame to the left edge of the character. The b-space is the width of the character. The c-space is the distance from the right edge of the character to the right of the character frame. Negative values of a and c allow adjacent character frames to overlap. See also *character increment* , and *space default values* .

average char width - The average horizontal distance from the left edge of one character to the left edge of the next. Contrast with *max char increment* .

baseline - The line on which the bottom of a character rests, and below which a descender extends.

beak char code point - The *code point* of the space or break character. Contrast with *default char code point* , *first char code point* , and *last char code point* .

character increment - A set of three values (*a-space* , *b-space* , and *c-space*) that define the proportions of a character. The sum of the three values (a+b+c) specifies only one value for the entire character increment. See also *font width* and *space default values* .

character rotation - The angle by which each character is rotated around its own center, increasing clockwise from vertical. Contrast with *character slope* and *inline direction* .

character slope - The angle by which a character is slanted, increasing clockwise from vertical. Contrast with *character rotation* and *inline direction* .

default char code point - The *code point* of the character to be used if a *code point* outside the range of a font is passed to an application using that font. Contrast with *break char code point* , *first char code point* , and *last char code point* .

em height - The maximum distance above the *baseline* reached by an uppercase symbol. Contrast with *x height* .

external leading - The vertical distance from the bottom of one character to the top of the character below it. Contrast with *internal leading* and *max baseline extent* .

first char code point - The *code point* of the first character. All numbers between the *first char code point* and the *last char code point* must represent a character in the font. Contrast with *break char code point* , *default char code point* , and *last char code point* .

fixed spacing - The same amount of space separates each character. Contrast with *proportional spacing* .

font weight - The line-thickness of a character relative to its size. Contrast with *font width* .

font width - The relative width of a character to its height; condensed fonts are very narrow while expanded fonts are very wide. See also *character increment* . Contrast with *font weight* .

inline direction - The angle of a line of type, increasing clockwise from horizontal. Contrast with *character rotation* and *character slope* .

internal leading - The vertical distance from the top or bottom of a character to any accent marks that may appear with it. Contrast with *external leading* .

last char code point - The *code point* of the last character. All numbers between the *first char code point* and the *last char code point* must represent a character in the font. Contrast with *break char code point* , *default char code point* , and *first char code point* .

lowercase ascent - The maximum distance above the *baseline* reached by any part of any lowercase character. Contrast with *maximum ascender* and *x height* .

lowercase descent - The maximum distance below the *baseline* reached by any part of any lowercase character. Contrast with *maximum descender* .

max baseline extent - The maximum space occupied by the font (typically, the sum of the *maximum ascender* and *maximum descender*). Contrast with *external leading* and *max char increment* .

max char increment - The maximum horizontal distance from the left edge of one character to the left edge of the next character to the right. Contrast with *average char width* and *max baseline extent* .

maximum ascender - The maximum distance that any part of any character may extend above the *x height* of a font. Contrast with *lowercase ascent* and *maximum descender* .

maximum descender - The maximum distance that any part of any character may extend below the *x height* of a font. Contrast with *lowercase descent* and *maximum ascender* .

maximum vert point size - The maximum vertical dimensions to which a font can be resized. Contrast with *minimum vert point size* and *nominal vert point size* .

minimum vert point size - The minimum vertical dimensions to which a font can be resized. Contrast with *maximum vert point size* and *nominal vert point size* .

nominal vert point size - The normal display size of a font. Contrast with *maximum vert point size* and *minimum vert point size* .

pel - The smallest element of a display surface that can be independently assigned color and density.

point - Printer's unit of measurement. There are 72 points to an inch (approximately 3.5 points to a millimeter).

proportional spacing - The space that each character occupies is in proportion to its width. See also *font width* . Contrast with *fixed spacing* .

Registry ID - A code number that Presentation Manager uses to register a font file as a resource.

space default values - Values that specify the space to be left between characters. Once defined, they are used for the entire font, and do not have to be specified for each character. However, they can be changed for characters that require more or less spacing than the defaults provide, by giving values for the *a Space* and the *c Space* . See also *character increment* .

strikeout position - The distance of the strikeout character above the *baseline* (in *pels*). See also *strikeout size* and *underscore position* .

strikeout size - The size of the strikeout character (in *points*). See also *strikeout position* and *underscore size* .

subscript position - The distance of a subscript character of a font below the *baseline* (in *pels*). See also *subscript size* and *superscript position* .

subscript size - The size of a subscript character (in *points*). See also *subscript position* and *superscript size* .

superscript position - The distance of a superscript character above the *baseline* (in *pels*). See also *subscript position* and *superscript size* .

superscript size - The size of a superscript character (in *points*). See also *subscript size* and *superscript position* .

target dev resolution X - The number of *pels* per inch in the horizontal axis of a display device on which a font is to be displayed. Contrast with *target dev resolution Y* .

target dev resolution Y - The number of *pels* per inch in the vertical axis of a display device on which a font is to be displayed. Contrast with *target dev resolution X* .

underscore position - The distance in *pels* of the first underscore stroke from the *baseline* of a font. Successive strokes below this create a heavier underscore. See also *strikeout position* and *underscore size* .

underscore size - The size of the underscore character measured in single strikeout strokes. See also *strikeout size* and *underscore position* .

x height - The maximum distance above the *baseline* reached by a lowercase character. Contrast with *em height* and *lowercase ascent* .

Graphics Attributes

Every graphics presentation space has a set of *graphics attributes* . A normal presentation space has a larger set of graphics attributes than a micro or cached micro presentation space. (Segment-related attributes, for example, do not apply in micro presentation spaces.) These attributes all have default values, which means that they always have an effect on graphics, even if you have not explicitly specified their values. The attributes can be broken into two general groups. The first group comprises those attributes that form a part of the picture and that can vary as the picture is drawn. These are:

- All primitive attributes
- The segment attributes
- The primitive tag
- The current position
- The viewing window
- The clipping path
- The model and segment transformations
- The viewing transformation.

With the exception of the segment attributes and the viewing transformation, these attributes are reset to their default values at the start of a root segment, unless the fast-chaining attribute is set. If the fast-chaining attribute is set, their values cannot be guaranteed. You should, therefore, explicitly set any attribute values required in the segment. Similarly, the values of these attributes cannot be guaranteed following a call to:

Any GpiDraw... function

Any GpiCorrelate... function
GpiCallSegmentMatrix
GpiCloseSegment

Therefore, if any of these functions is followed by primitives outside a segment, you should explicitly set required attribute values. When GpiCloseSegment is followed by GpiOpenSegment (and also between any two segments in the chain), the fast-chaining attribute determines what happens to the current values of these attributes. The viewing transformation and the segment attributes are unaffected by fast-chaining.

These attributes take effect when graphics are sent to an output device, *not* when graphics are defined. For this reason, the calls to GpiQuery... that retrieve the current values of these attributes are invalid in retain mode. The majority of the group-one attribute-setting functions cause graphics orders to be added to the current segment.

The second group of attributes defines the environment in which the picture is drawn. These attributes do not normally vary as the picture is drawn, but have an overall effect on the result of any drawing or correlation operation. This group includes:

- The default viewing transformation
- The page viewport
- The graphics field
- The clipping region
- The pick-aperture size
- The drawing controls.

The functions to GpiQuery... that retrieve the current values of these attribute are valid in all drawing modes. None of the group-two attribute-setting functions causes graphics orders to be added to the current segment.

The following table lists the graphics attributes, identifies the GPI function or functions that you use to change current settings, and lists the default value of each one. Note that these default values are the initial settings. The table also indicates whether the attribute is a group-one or a group-two attribute.

Graphics Attribute	GPI Function	Default Value	Group One	Group Two
Arc parameters	GpiSetArcParams	(1,1,0,0)	Yes	
Foreground color	GpiSetColor(1)	CLR_DEFAULT (device-dependent)	Yes	
Foreground mix	GpiSetMix(1)	FM_DEFAULT (overpaint)	Yes	
Background color	GpiSetBackColor(1)	CLR_DEFAULT (device-dependent)	Yes	
Background mix	GpiSetBackMix(1)	BM_DEFAULT (leave-alone)	Yes	
Character angle	GpiSetCharAngle(1)	(1,0)	Yes	
Character box	GpiSetCharBox(1)	Device-dependent	Yes	
Character Direction	GpiSetCharDirection(1)	CHDIRN_DEFAULT (left to right)	Yes	
Character mode	GpiSetCharMode(1)	CM_DEFAULT (mode 1)	Yes	
Character set	GpiSetCharSet(1)	LCID_DEFAULT (system font)	Yes	
Character shear	GpiSetCharShear(1)	(0,1)	Yes	
Line end	GpiSetLineEnd(1)	LINEEND_DEFAULT (flat)	Yes	
Line join	GpiSetLineJoin(1)	LINEJOIN_DEFAULT (beveled)	Yes	
Line type	GpiSetLineType(1)	LINETYPE_DEFAULT (solid)	Yes	
Line width	GpiSetLineWidth(1)	LINEWIDTH_DEFAULT (1.0)	Yes	
Geometric line width	GpiSetLineWidthGeom(1)	1	Yes	
Marker	GpiSetMarker(1)	MARKSYM_DEFAULT (cross)	Yes	

Marker box	GpiSetMarkerBox(1)	Device-dependent	Yes	
Marker set	GpiSetMarkerSet(1)	LCID_DEFAULT (system marker set)	Yes	
Pattern	GpiSetPattern(1)	PATSYM_DEFAULT (solid)	Yes	
Pattern reference point	GpiSetPatternRefPoint(1)	(0,0)	Yes	
Pattern set	GpiSetPatternSet(1)	LCID_DEFAULT (system pattern set)	Yes	
Model transformation	GpiSetModelTransformMatrix	Identity	Yes	
Instance transformation	GpiCallSegmentMatrix	Identity	Yes	
Segment transformation	GpiSetSegmentTransformMatrix	Identity	Yes	
Viewing transformation	GpiSetViewingTransformMatrix	Identity	Yes	
Default viewing transformation	GpiSetDefaultViewMatrix	Identity		Yes
Page viewport	GpiSetPageViewport	Device-dependent		Yes
Clipping path	GpiSetClipPath	No clipping	Yes	
Viewing window	GpiSetViewingLimits	No clipping	Yes	
Graphics field	GpiSetGraphicsField	No clipping		Yes
Clipping region	GpiSetClipRegion	No clipping		Yes
Tag	GpiSetTag	0	Yes	
Segment attributes	GpiSetInitialSegmentAttrs GpiSetSegmentAttrs	ATTR_DETECTABLE - OFF ATTR_VISIBLE - ON ATTR_CHAINED - ON ATTR_DYNAMIC - OFF ATTR_FASTCHAIN - ON ATTR_PROP_DETECTABLE - ON ATTR_PROP_VISIBLE - ON	Yes	
Pick-aperture size	GpiSetPickAperturePosition	Device-dependent		Yes
Current position	GpiSetCurrentPosition GpiMove[2]	(0,0)	Yes	
Drawing controls	GpiSetDrawControl	DCTL_ERASE - OFF DCTL_DISPLAY - ON DCTL_BOUNDARY - OFF DCTL_DYNAMIC - OFF DCTL_CORRELATE - OFF		Yes

Notes:
(1). Can also be set using GpiSetAttrs.
(2). Is also updated indirectly by most primitive-drawing functions.

Each of the following functions causes all group-one and group-two graphics attributes to be set to their default values:

- GpiCreatePS
- GpiSetPS
- GpiResetPS, if GRES_ALL or GRES_SEGMENTS is specified
- WinGetPS
- WinGetScreenPS
- WinBeginPaint, with a NULL presentation space handle.

In specific circumstances, some of the GPI functions modify the group-one attributes and thus make their values unpredictable. Therefore, when you call any of these functions, you should respecify attribute values that have a particular importance to your application. For example, if the current foreground color is CLR_RED before you call GpiDrawChain you cannot always rely on the current color remaining CLR_RED when GpiDrawChain completes. If you want to continue working in red, respecify the color when GpiDrawChain completes. In general, the functions that affect group-one-attribute values are those related to the drawing and correlation of retained graphics, and to the creation, closing, and deletion of graphics segments.

With the exception of the viewing transformation and the segment attributes, the default values of these attributes apply to all primitive-drawing that occurs outside a segment bracket.

- The following function sets the group-one attributes to their default values, and in addition sets the current clipping region and page viewport to their default values.

 - GpiAssociate.

- Each of the following functions makes the group-one-attribute values unpredictable:

 - GpiCloseSegment
 - GpiCorrelateChain
 - GpiCorrelateFrom
 - GpiCorrelateSegment
 - GpiErase
 - GpiCallSegmentMatrix
 - GpiDrawDynamics
 - GpiRemoveDynamics.

 Note: In the last two functions here, the foreground-mix value is always set to FM_XOR, and the background-mix value is always set to BM_LEAVEALONE.

- Each of the following functions makes group-one-attribute values unpredictable if a segment to be deleted is open when the function is called:

 - GpiDeleteSegments
 - GpiDeleteSegment

- Each of the following functions makes group-one-attribute values unpredictable if there is no open segment when the function is called:

 - GpiDrawChain
 - GpiDrawFrom
 - GpiDrawSegment.

 If there is an open segment when any of these functions is called, and that segment was the last one drawn, then group-one-attribute values are unaffected. If, however, dynamic segments were caused to be redrawn by the same function, group-one-attribute values are made unpredictable. This occurs because dynamic segments are always drawn after nondynamic segments.

Line and Arc Primitives

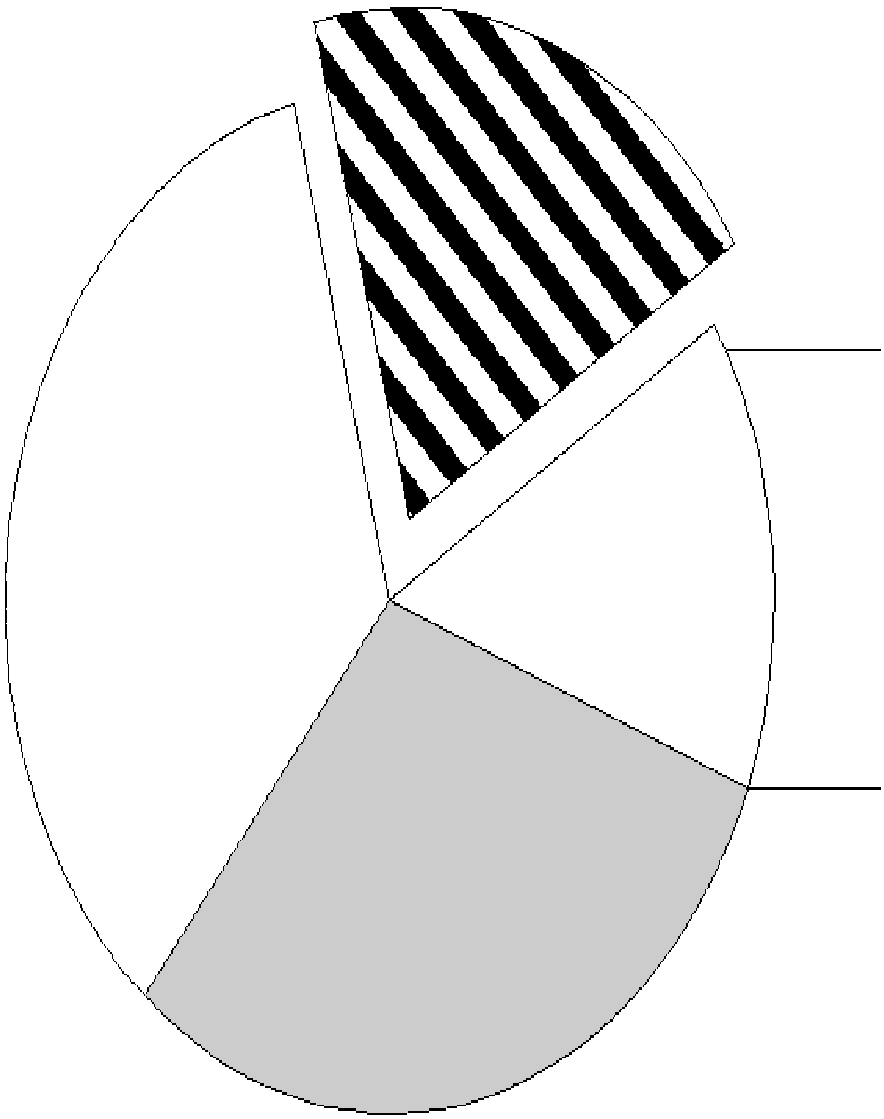
Line and arc primitives are graphics building blocks for creating pictures that consist of objects such as polygons, circles, fillets, ellipses, and other geometric figures.

The following topics are related to information in this chapter:

- Presentation spaces and device contexts
- Color and mix attributes
- Area primitives

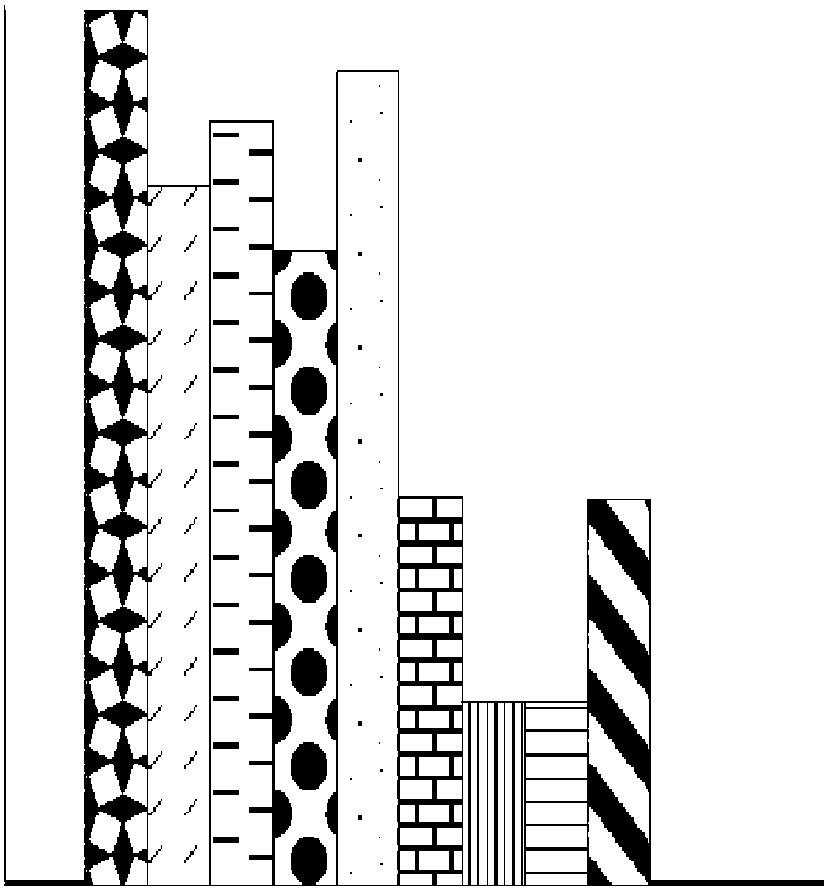
About Line and Arc Primitives

Simple drawing applications use line and arc primitives as drawing tools. They are useful, for example, in spreadsheet applications for constructing pie charts, bar charts, and graphs.



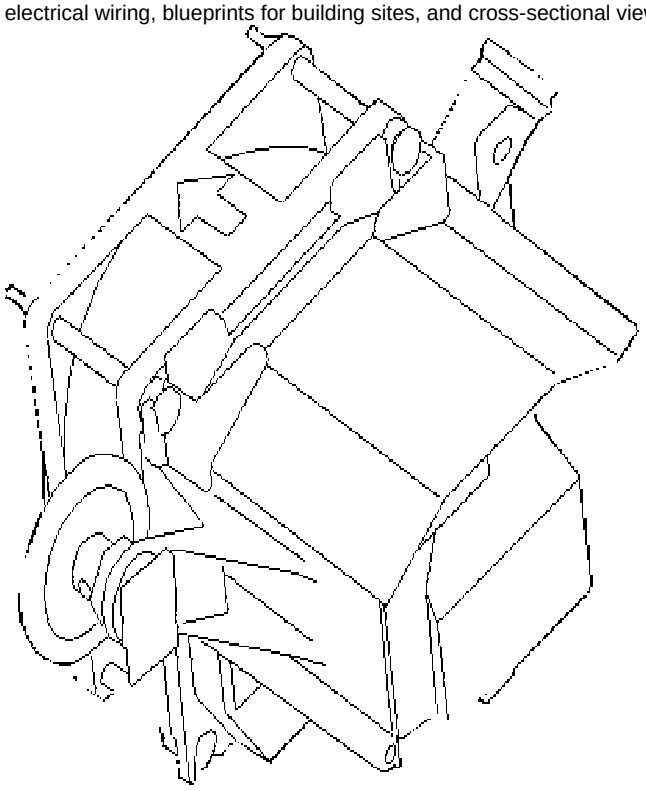
Sample Pie Chart Created with Line and Arc Primitives

Following are more illustrations that were drawn using line and arc primitives.



Sample Bar Graph Created with Line and Arc Primitives

Computer-aided-design (CAD) applications combine line and arc primitives to draw such complex pictures as schematic diagrams for electrical wiring, blueprints for building sites, and cross-sectional views of machinery.



Sample Blueprint Created with Line and Arc Primitives

Line and arc primitives are actually two families of primitives that contain many variations of straight lines and curved lines, respectively. They are presented together in this chapter because all the variations are governed by the attributes found in the data structure LINEBUNDLE.

An application draws the line and arc primitives by first calling GpiMove or GpiSetCurrentPosition, either of which sets the current position to a specified starting point. GpiMove ignores the AM_PRESERVE mode, whereas GpiSetCurrentPosition saves the current position if the AM_PRESERVE mode is set. The more sophisticated function, GpiPolyLineDisjoint, contains its own starting point and does not need to be preceded by GpiMove, or GpiSetCurrentPosition.

Prior to calling a line or arc function, the current position can be determined with GpiQueryCurrentPosition.

Attributes of Line and Arc Primitives

The attributes of the line and arc primitives contained in LINEBUNDLE are:

- Line width
- Geometric width
- Line type
- Line end
- Line join
- Line color
- Line mix

The line end and line join attributes apply only to lines drawn in a primitive called a *path* .

When an application creates a presentation space, the line and arc attributes are set to the default values shown in the following table.

Line Attribute Default Values

Attribute	Default Value	Function that Redefines Attribute
Width	1.0	GpiSetLineWidth
Geometric width	None	GpiSetLineWidthGeom
Type	LINETYPE_SOLID	GpiSetLineType
Color	CLR_NEUTRAL	GpiSetAttrs (LBB_COLOR)
Mix attribute	FM_OVERPAINT	GpiSetAttrs (LBB_MIX_MODE)

Line Width and Geometric Width

The operating system defines two types of lines: cosmetic and geometric, representing two separate concepts.

Cosmetic lines represent the mathematical ideal of a line being an entity with only one dimension-length. When cosmetic lines are drawn, they usually are only one pel wide.

The line width attribute provides a visual method of distinguishing different types of lines. For example, on a map, roads might be drawn thicker than railroad tracks. The line width attribute defines a multiplier to be applied to the normal (or default) line width. Your application can determine the cosmetic line width by calling GpiQueryLineWidth.

There are certain devices that do not define the thickness of cosmetic lines in terms of a number of pels. The PM interface interprets the specified width for these devices as shown in the following table:

Identifer	Description
LINEWIDTH_DEFAULT	The default width for the device.
LINEWIDTH_NORMAL	The line width equivalent to a multiplier of 1. Identical to LINEWIDTH_DEFAULT unless changed with GpiSetDefAttrs.
LINEWIDTH_THICK	The line width equivalent to a multiplier greater than 1.

Your application can set the cosmetic line width with either a multiplier or an identifier value using GpiSetLineWidth. Since the line represents a widthless ideal, the actual drawn width is a fixed value. The line width remains unchanged when you increase the size of, or zoom in on, a graphics object. The cosmetic line width is not subject to transformations.

In contrast, *geometric lines* represent an area whose thickness is equal to the specified line width. Geometric lines are subject to changes in scale through transformations, in the same manner as geometric figures.

The width of a geometric line does not have a default value. Width is treated as a line attribute for programming convenience, but it actually is a geometric property. A line width is set with GpiSetLineWidthGeom. If a width is already specified, that width can be determined with GpiQueryLineWidthGeom.

To use geometric line width, the lines are assembled into a path. When you use a path to define wide lines, you can specify the following:

- Geometric width for the lines, using the geometric line width attribute
- Style of the line ends, using the line end attribute
- Style of the junctions of the lines, using the line join attribute

These three attributes apply only to geometric lines drawn using paths.

Line Types

Line type , also called *line style* , defines the way a line or arc is drawn: as a solid line, a series of dashes, a series of dots, or a combination of dashes and dots.

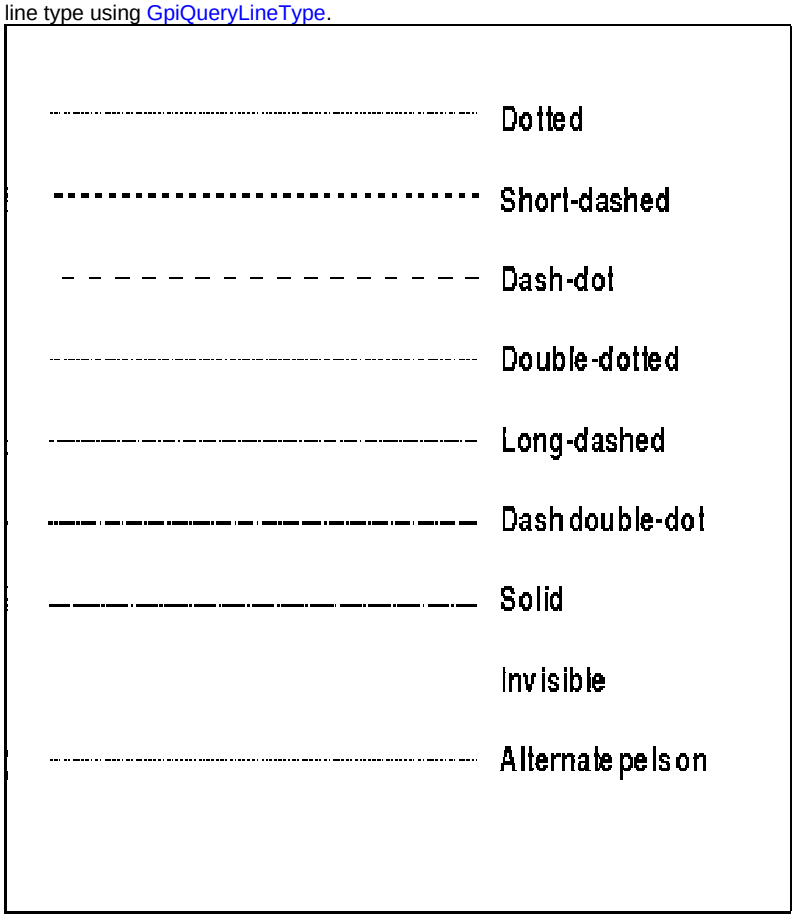
The line type for all subsequent line primitives is selected using GpiSetLineType. If you want to change the line type to LINETYPE_DOT, for example, any line or arc primitive subsequently drawn is drawn as a dotted line. The following table illustrates the nine standard line types provided by PM.

You cannot define other line types for PM applications.

Standard Line Types

Type	Identifer	Long Value
Dotted line	LINETYPE_DOT	1L
Short-dashed line	LINETYPE_SHORTDASH	2L
Dash-dot line	LINETYPE_DASHDOT	3L
Double-dotted line	LINETYPE_DOUBLEDOT	4L
Long-dashed line	LINETYPE_LONGDASH	5L
Dash-double-dot line	LINETYPE_DASHDOUBLEDOT	6L
Solid line	LINETYPE_SOLID	7L
Invisible line	LINETYPE_INVISIBLE	8L
Alternate pels on	LINETYPE_ALTERNATE	9L

The default line type (LINETYPE_DEFAULT) is identical to the LINETYPE_SOLID type, and has a long value of 0L. The error line type (LINETYPE_ERROR) has a long value of -1L. The following figure illustrates the nine line types. Your application can determine the current line type using GpiQueryLineType.



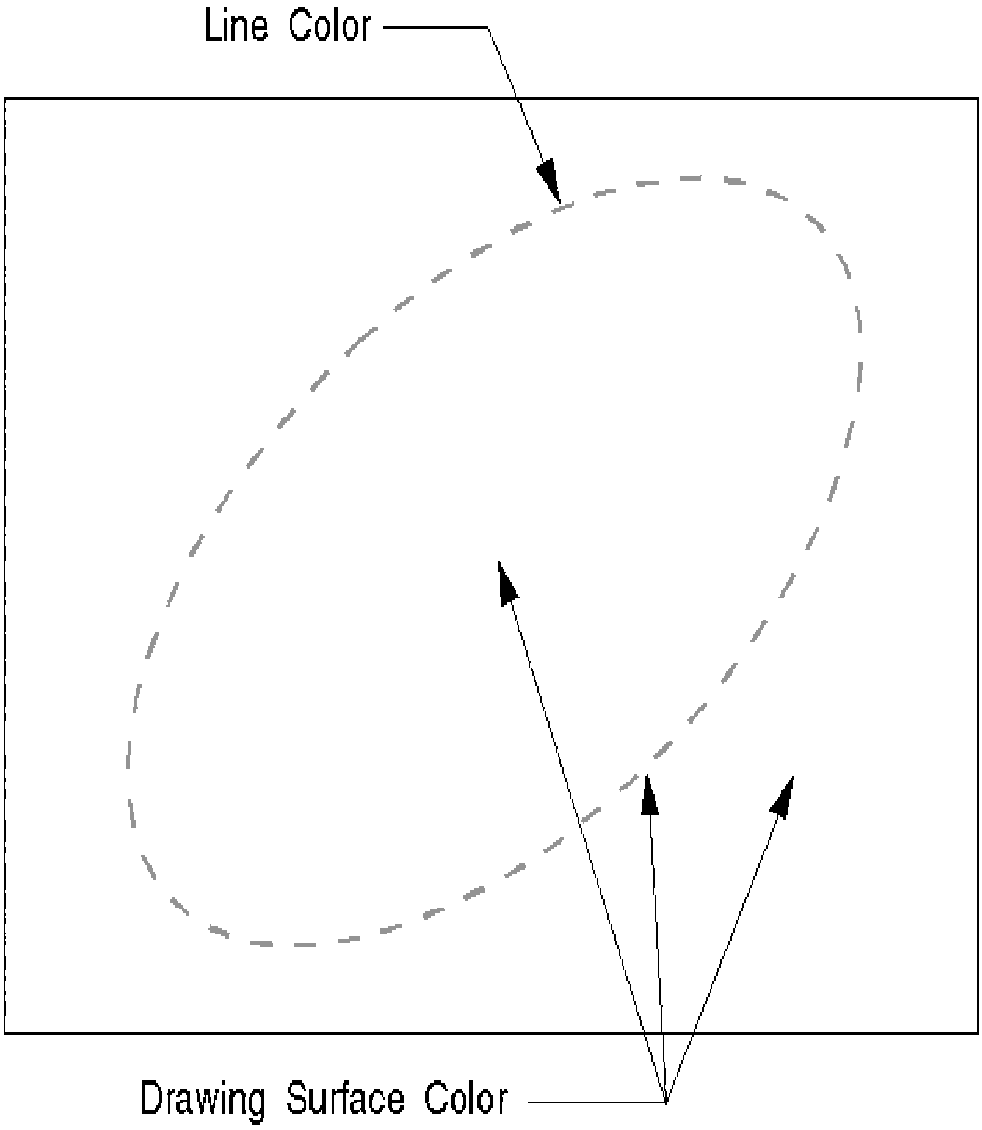
The Nine Line Types

Line Color and Mix Attributes

The color attribute defines the color used to draw a primitive or an object. The mix attribute determines how the color of a primitive or an object is combined with the color of the drawing surface, or any other objects on the surface.

The line color defines the color used to draw the output from any of the operating system's line functions. When a presentation space is created, the line color initial default is black. Unlike certain other primitives, the line and arc primitives do not have a background color. The current color of the drawing surface automatically plays a greater role in the appearance of line and arc primitives than in those primitives that do have a background color. Specifically, when an application draws a dotted or dashed line, the color that appears between the dots or dashes is the current drawing-surface color, as shown in the following figure.

Line and arc primitives have only a color attribute for the actual line. The mix attribute controls the combination of line color with drawing surface color.



Line and Arc Primitives

When a presentation space is created, the line mix attribute initial default is FM_OVERPAINT. The *overpaint* mix attribute specifies that the line color is not modified by the color of the drawing surface. If the line mix attribute is changed, the line color is mixed with colors that are already on the drawing surface.

To specify a new color or mix attribute use GpiSetAttrs. This function accepts as input the type of primitive, for example PRIM_LINE, a list of attributes that are to be changed, a list of attributes that are to be set to their default values, and the values for the attributes that are to be changed. GpiSetAttrs is useful for specifying colors and mix attributes just for a specific data structure (for example, the LINEBUNDLE structure). GpiSetAttrs also provides some protection against invalid colors.

To determine the current line color and mix attribute, use GpiQueryAttrs. This function accepts as input the primitive type and the attributes in question. It returns as output an array of values for the specifically queried attributes.

To reset the default line color and mix attribute, just as with any other attribute specified in the LINEBUNDLE data structure, use GpiSetDefAttrs. This function accepts as input the type of primitive, for example PRIM_LINE, the attributes to be changed, and the values that will become the new default values. The changing of default values is important when working with segments. Changing the default values during a series of drawing functions is not recommended.

The line color and mix attribute also can be specified with GpiSetColor and GpiSetMix, respectively. However, those two functions have the disadvantage of specifying the color and mix attribute for all primitive BUNDLE data structures that have a component for foreground color and foreground mix attribute. The queries, GpiQueryColor and GpiQueryMix, determine the color and mix attribute as specified by GpiSetColor and GpiSetMix. If the line color or mix attribute is specified individually, the aforementioned queries can return a value inconsistent with the current line color or mix attribute.

Line Primitive Family

The following table describes the three variants of the basic line primitive and the functions that draw them.

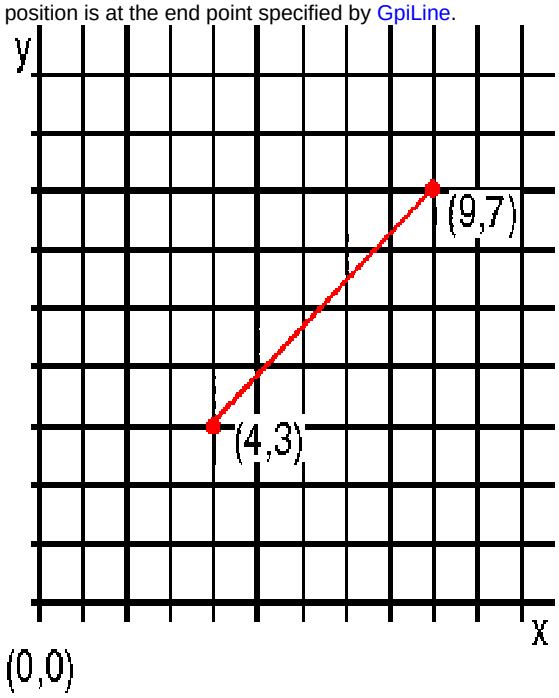
Functions that Draw Straight Lines

Variants	Function	Description
Lines	GpiLine	Draws a single line from the current position to a specified point.
Polylines	GpiPolyLine	Draws a series of connected lines, from the current position through successive points.
(series of lines)	GpiPolyLineDisjoint	Draws a series of unconnected lines.
Boxes	GpiBox	Draws a rectangular box with one corner at the current position.

When the operating system draws a line, it includes the pels at the starting and ending points of the line. The algorithm used to draw the rest of the line depends on the device driver. For example, a driver for a raster device might use a modified Bresenham algorithm to draw a line, but a driver for a vector device, such as a plotter, simply would connect the starting and ending points of the line. In all cases, the result is a line primitive that looks the same from device to device.

Lines

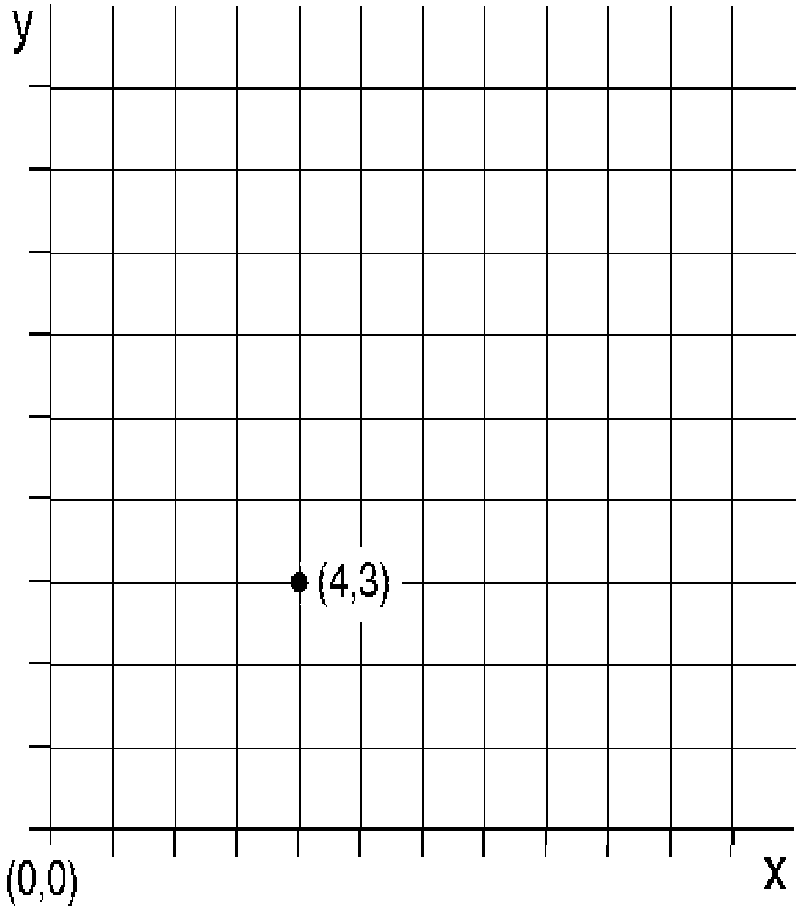
GpiLine draws a line from the current position to a specified end point, as shown in the following figure. After drawing the line, the current position is at the end point specified by GpiLine.



GpiLine

The current position is (4,3), and the specified end point of GpiLine is (9,7). When the line has been drawn, (9,7) becomes the current position.

To draw a single point, you can use GpiLine with an end point identical to the current position, as shown in the following figure. The current position can be determined using GpiQueryCurrentPosition.

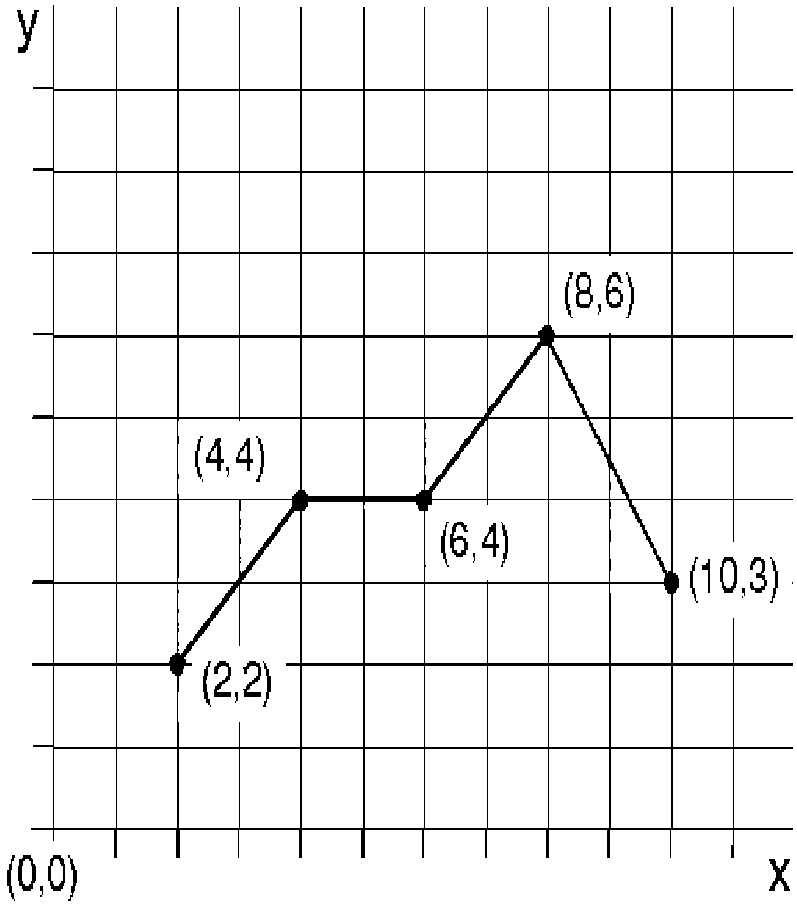


GpiLine Used to Draw a Single Point

Polylines

GpiPolyLine draws a sequence of connected lines, starting at the current position and passing through a series of specified coordinate positions, as shown in the following figure. After drawing the series of lines, the current position is at the end point of the last line specified by GpiPolyLine.

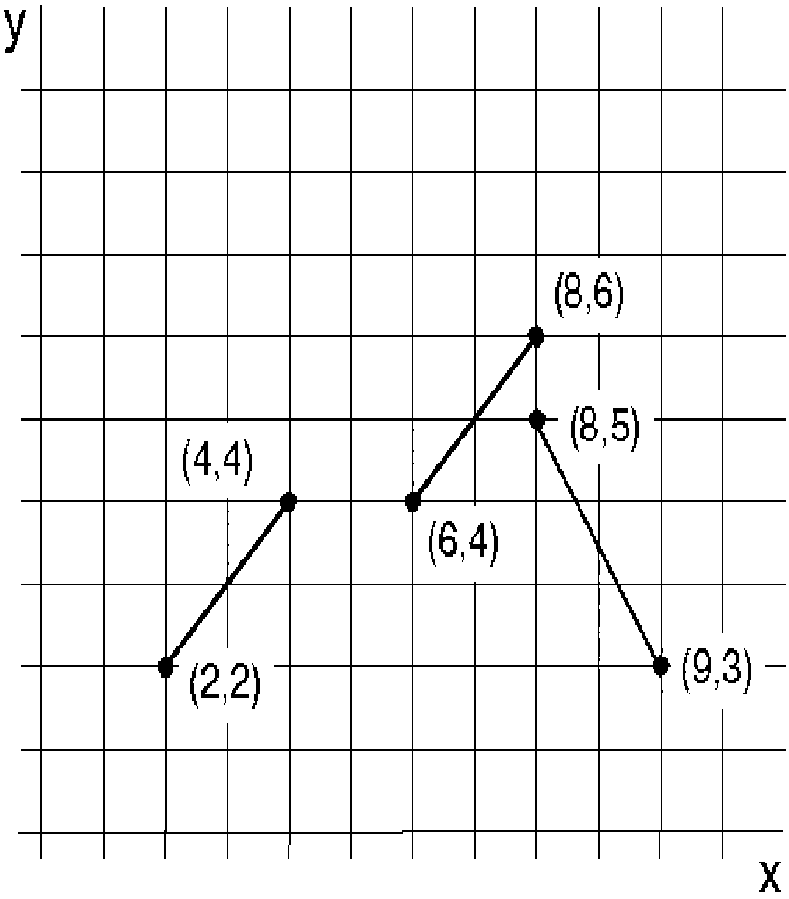
If you are drawing a graph that has five connected lines, you can use GpiPolyLine once rather than GpiLine five times. GpiPolyLine accepts as input a number of points and an array of point coordinates.



GpiPolyLine

The starting position is (2,2), and the polyline is drawn through (4,4), (6,4), (8,6), and (10,3). The new current position is (10,3).

GpiPolyLineDisjoint is a new function that eliminates having to use a series of GpiMove and GpiLine functions to draw multiple lines that are not connected. GpiPolyLineDisjoint accepts as input an even number, the number of points, and an array of point coordinates. The first point in a point-pair is the starting point; the second, the end point of that line segment. Upon completion, the end point of the final line becomes the current position. The following figure shows the results of a GpiPolyLineDisjoint call.



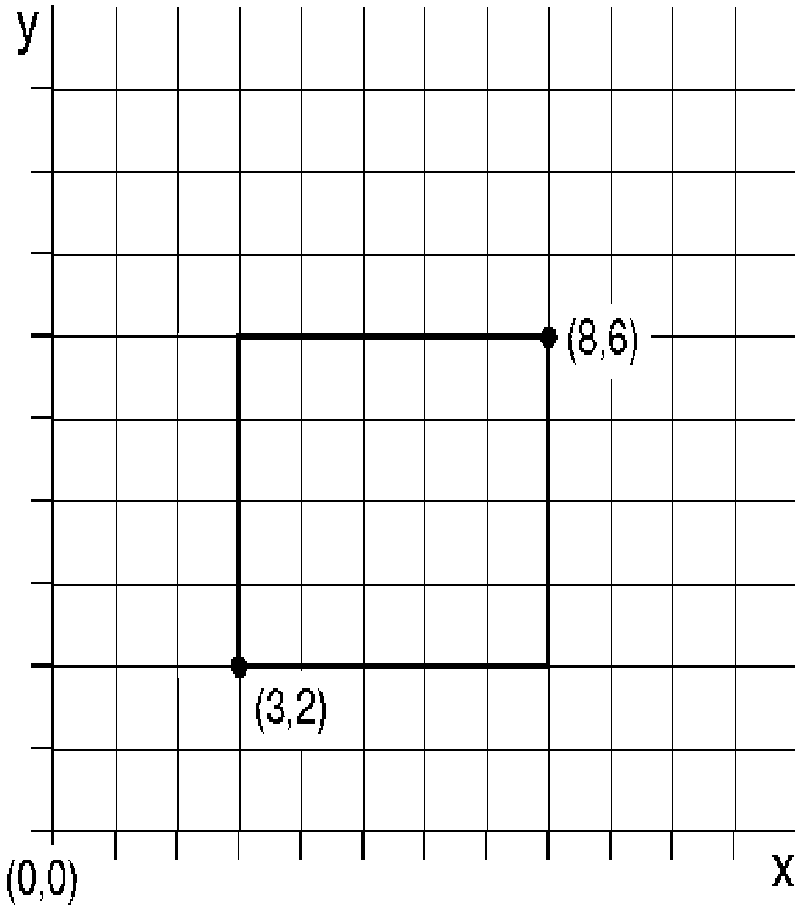
GpiPolyLineDisjoint

A graph with discontinuities.

Boxes

At its simplest, GpiBox draws a rectangular box with one corner at the current position and the diagonally-opposite corner at a position that you specify. The sides of the box are parallel to the x- and y-axes. Like GpiPolyLine, GpiBox lets you draw a number of connected lines using a single function rather than four separate GpiLine, functions. The current position is unchanged by GpiBox.

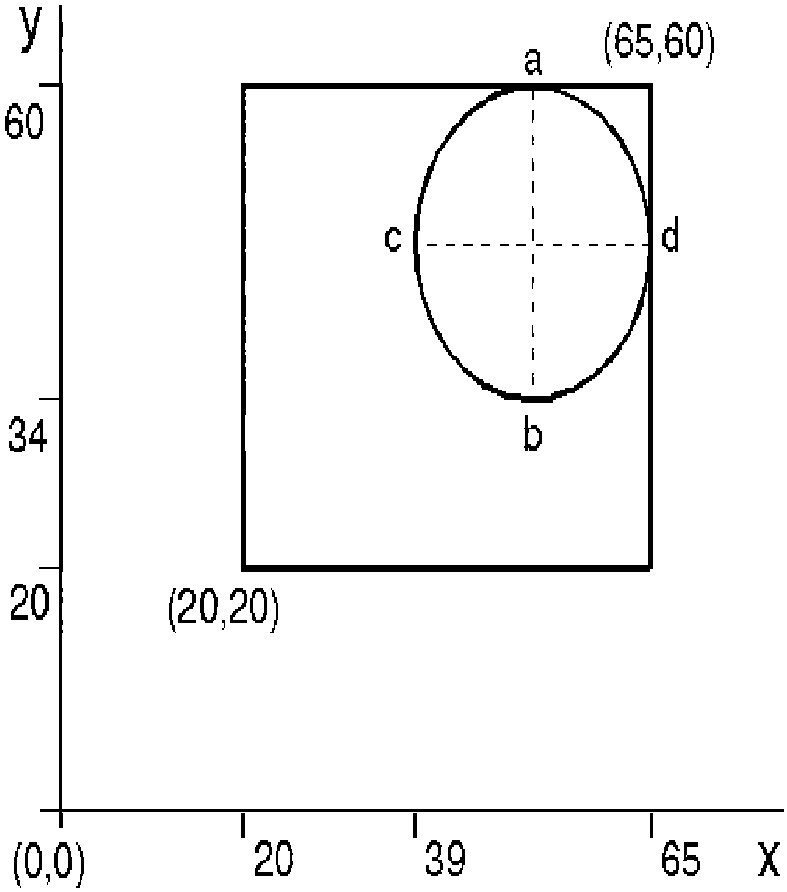
Note: The start and end position of any closed shape are always the same. Therefore the current position is unchanged after drawing a closed figure. The GpiBox primitive is shown in the following figure.



The Box

The current position is (3,2) and the corner position is at (8,6). When the box has been drawn, the current position remains at (3,2).

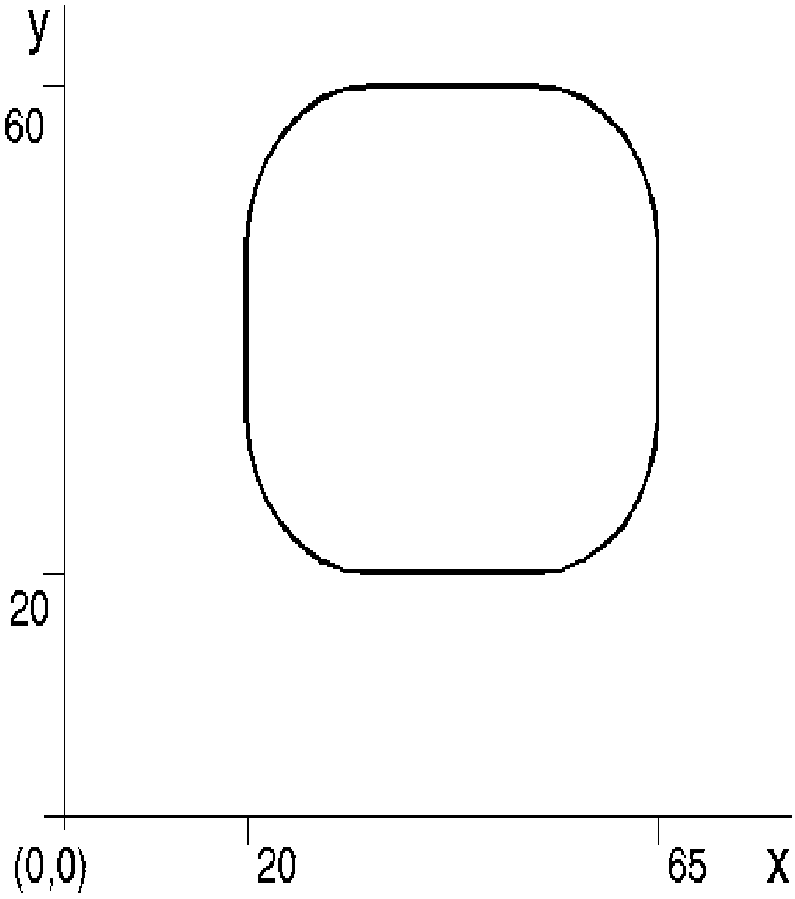
In addition to the corner position, GpiBox accepts as input an option for rounded corners and for filled interior. The PM programming interface rounds the corners of a box by drawing an elliptical section in place of the square corner. Two GpiBox parameters, */lHRound* and */lVRound*, represent the horizontal and vertical length of the full axis of the ellipse used to round each corner. If the two values are equal, a quarter-circle is used for the rounding. If the two values are 0, no rounding is performed. The following figure shows an example of rounding corners using a quarter-circle.



Quarter-Circle Box-Corner Rounding

The current position is (20,20). GpiBox is called with a corner position of (65,60) and an identical vertical and horizontal rounding value of 26.

The following figure shows the complete result. The intermediate steps are not visible to the user of your application.



Box with Rounded Corners

All four corners are rounded with identical values.

Since GpiBox can be used to define a closed figure, it also accepts as input a long value signifying a filled interior. The long value */Control/* can be any of the following:

Value	Description
DRO_OUTLINE	Draw the box only.
DRO_FILL	Fill the box interior only.
DRO_OUTLINEFILL	Draw the box and fill its interior.

The pattern that fills the interior and other drawing options are controlled by the data structure AREABUNDLE. GpiFullArc is the only other line and arc primitive that can be used to define a closed figure.

Attributes of Arc Primitives

The arc primitive shares width, type, and color and mix attributes with line primitives. For example, if you use GpiSetLineType to change the line type to LINETYPE_DASHDOT, all subsequent arcs are drawn with a dash-dot line. In addition to the line attributes defined in the LINEBUNDLE data structure, arc primitives in the simple-arc family are influenced by the values in the ARCPARAMS data structure. Arc primitives in the multiple-arc family have a different method of construction and are not influenced by ARCPARAMS.

In terms of geometrical pictures, the simple arcs contain full or partial:

Circles	Closed curves whose center is equidistant from every point on the curve.
Ellipses	Closed curves defined by two fixed points such that the sum of the distances from any point on the curve

to the two fixed points is constant.

Multiple arcs contain:

Fillets	Curves that are tangential to the two lines defined by three control points.
Splines	Curves that, given four control points, are tangent to the first and last of three intersecting lines.

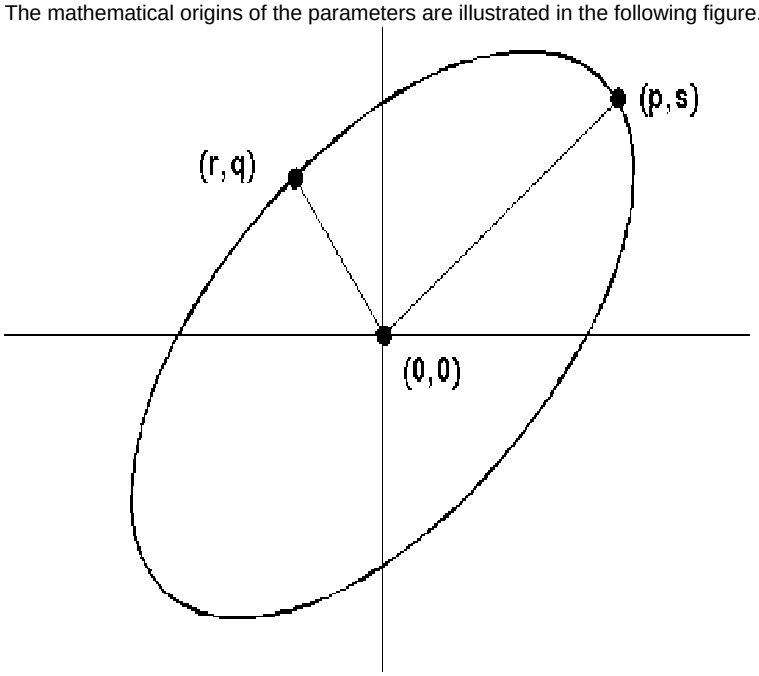
There are three simple arc operations that begin with a *unit circle* that lies at the origin of world coordinate space. This unit circle defines the *current arc* on which subsequent full, partial, and 3-point arcs are based. Your application can define the following attributes of the current arc with GpiSetArcParams:

- Shape
- Orientation
- Size
- Drawing direction

GpiSetArcParams accepts as input an ARCPARAMS data structure that has four parameters (**p, q, r**, and **s**). Of the four:

- **p** scales in the x-direction
- **q** scales in the y-direction
- **r** and **s** are shear components

The mathematical origins of the parameters are illustrated in the following figure.



The ARCPARAMS Values in World Coordinates

The values were derived from the major and minor axis of an ellipse.

An application can determine the current arc parameters with GpiSetDefArcParams which copies the current arc parameters to their corresponding fields in the supplied ARCPARAMS structure. The application can set the arc parameters with GpiSetArcParams. This function accepts as input a copy of the ARCPARAMS structure that contains the new arc parameters. The default values of **p, q, r**, and **s**, unless changed by GpiSetArcParams, define a unit circle:

- **p** and **q** are **1**
- **r** and **s** are **0**.

The arc parameters define a transformation that is applied to each point on the perimeter of the unit circle. For any point (x,y) on the perimeter of the unit circle, there exists a new point (x',y'), as determined by the following two linear equations:

```
x' = p x x + r x y
y' = s x x + q x y
```

These parameters form a 2-by-2 matrix,

```
p  r
s  q
```

that scales and shears simple arcs.

This transformation matrix is not related to the general *transformation functions* that move objects through *coordinate spaces*. It is a special purpose matrix that transforms the shape and size of the imaginary unit circle. The transformed unit circle, also called the *current arc*, is then used to define the shape and size of the simple arc functions in world coordinates.

After an arc has been described in world coordinates, it can be transformed with the transformation functions, just as any other primitive can.

A transformation is orthogonal when:

```
(p x r) + (s x q) = 0
```

If orthogonal, the line from the origin (0,0) to the point (p,s) is either:

- The radius of a circle
- Half the major or minor axis of an ellipse;

and, the line from the origin to the point (r,q) is either:

- The radius of a circle
- Half the minor or major axis of an ellipse.

An orthogonal transformation does not guarantee that the shape of an object as defined by an application will be the same as the shape of the object on the output device. For example, if the page units in the application are PU_PELS, and the pels on the device are rectangular (but not square), calling GpiFullArc produces an ellipse-not a circle- even when the arc parameters are set to their default values.

The product of the 2-by-2 matrix multiplication influences the direction of any arc based on the current arc, except a 3-point arc. The following table illustrates this directional influence.

If...	GpiFullArc and GpiPartialArc ...
(p x q) is greater than (r x s)	Draw the ellipse counterclockwise.
(p x q) is less than (r x s)	Draw the ellipse clockwise.
(p x q) equals (r x s)	Draw a straight line rather than an ellipse.

Simple-Arc Primitive Family

The following table describes the three variants of a simple-arc primitive and the functions that draw them. All are defined, to some extent, by the *current arc parameters*. As with line primitives, an application draws simple-arcs by first using GpiMove or GpiSetCurrentPosition to set the current position.

Functions that Draw Simple Arcs

Variants	Function	Description
Full arcs	GpiFullArc	Draws a circle or an ellipse.
Partial arcs	GpiPartialArc	Draws a straight line followed by a section of a circle or ellipse.

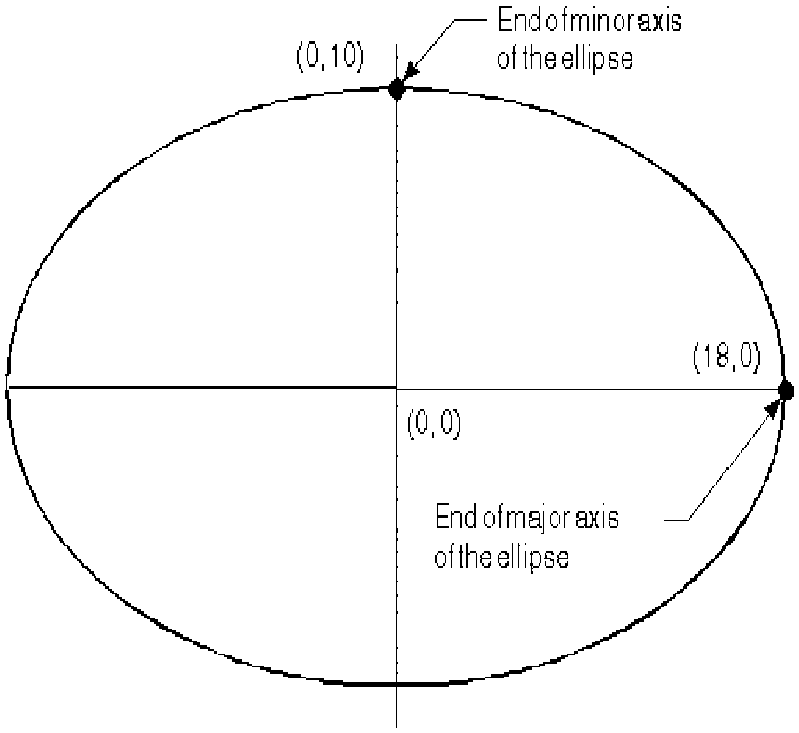
3-point arcs GpiPointArc Draws an arc through three points.

Full Arcs

GpiFullArc draws a complete circle or ellipse with its center at the current position. The current position remains unchanged. Whether GpiFullArc draws a circle or an ellipse depends on the current arc parameters. When the current arc is a circle, GpiFullArc draws a circle; when it is an ellipse, GpiFullArc draws an ellipse.

Defining an Ellipse

To define an ellipse as the current arc, use GpiSetArcParams where the values (**p,s**) and (**r,q**) are the world coordinates of the end points of the major and minor axes of the ellipse. For example, current arc parameters of (18,0) and (0,10) define an ellipse with a major axis of 36 coordinate units and a minor axis of 20 coordinate units. The resultant ellipse is shown in the following figure.



The Ellipse

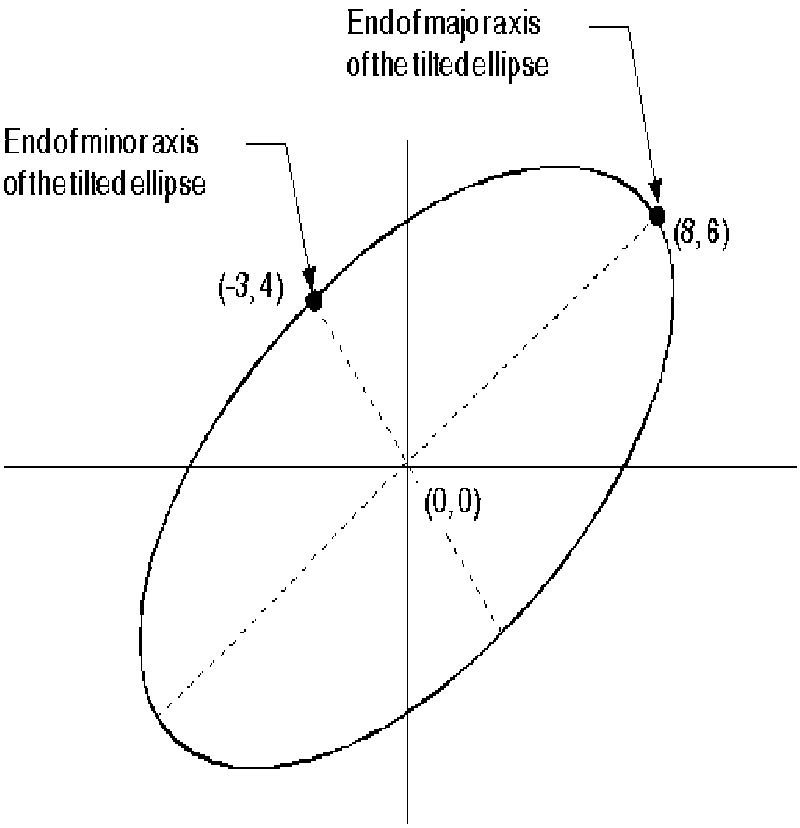
For maximum accuracy, create the axes of an ellipse so that they are at right-angles to each other. You can check this by ensuring that the following equation is always true:

$p \times r + s \times q = 0$

So, to check the above example:

0 x 18 + 10 x 0 = 0

You also can define a tilted ellipse as the current arc. None of the current arc parameters for a tilted ellipse will be 0, though you should still ensure that the axes of the ellipse are at right-angles to each other. The following figure shows a tilted ellipse defined with current arc parameters of (8,6) and (-3,4).



Tilted Ellipse

Defining a Circle

To define a circle as the current arc, (**r,q**) and (**p,s**) can be any such value that they lie on a circle. For example, if (**r,q**) and (**p,s**) are set to (-4, 3) (3,4), the current arc is a circle with a radius of 5 coordinate units. For simplicity, a circle can be defined by specifying current arc parameters where **p** is the radius of the circle, and $p = q$, $r = 0$, and $s = 0$. For example, to define a circle centered on the origin and with a radius of 10 world-coordinate units, use GpiSetArcParams with the values (0,10) (10,0).

The default values of the current arc parameters are (0,1) (1,0), which define a circle with a radius of one world-coordinate unit (a *unit* circle).

GpiFullArc Input Parameters
GpiFullArc accepts as input a multiplier value, so that the size of the full arc is increased or decreased in relation to the current arc. For example, if the current arc parameters define an ellipse whose major axis is 20 coordinate units and whose minor axis is 8 coordinate units, a multiplier of 2 in GpiFullArc creates an ellipse whose major axis is 40 coordinate units and whose minor axis is 16 coordinate units.

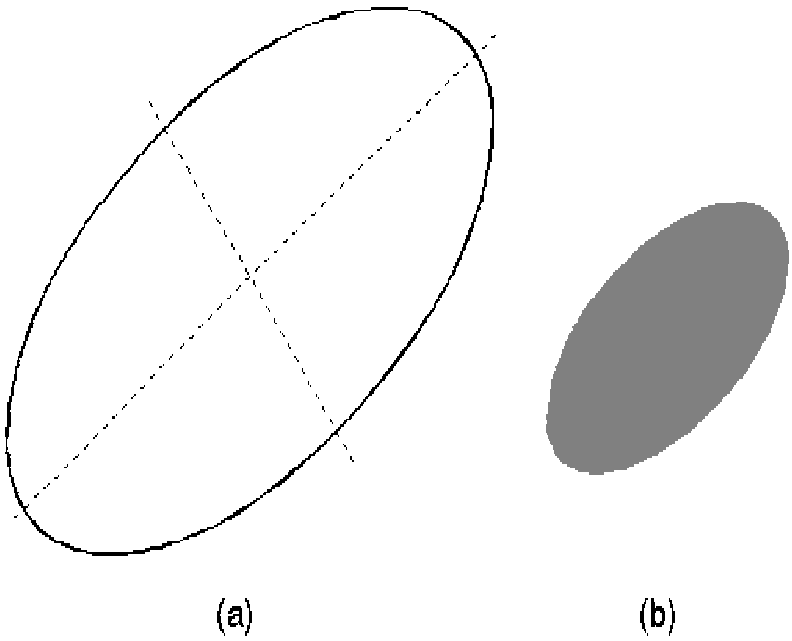
Because the arc parameters are integers when a fraction is required, for example when rotating an ellipse, greater precision can be obtained by scaling up the required arc parameter values as much as possible, then using a multiplier smaller than 1 to scale the ellipse back down to the required size.

With the default arc parameters defining a circle of 1 world coordinate unit, you can draw a circle of any size by allowing the arc parameters to default, then specifying the radius of the circle with the GpiFullArc multiplier. For example, to draw a circle with a radius of 12 coordinate units, call GpiFullArc with a multiplier of 12 and allow the GpiSetArcParams to default. Since GpiFullArc, like GpiBox, can be used to define

a closed figure, it also accepts as input a long value signifying a filled interior. The long value, *lControl*, can be:

Value	Description
DRO_OUTLINE	Draw the arc only.
DRO_FILL	Fill the arc interior only.
DRO_OUTLINEFILL	Draw the arc and fill its interior.

The pattern that fills the interior and other drawing options are controlled by the data structure AREABUNDLE. See Area and Polygon Primitives for detailed information on using areas.



The Full Arc

(a) is a tilted ellipse that is defined by the current arc parameters. (b) is drawn using GpiFullArc with:

- A multiplier value that reduces the size of the arc relative to the current arc
- An IControl value filling the interior with the current area-fill pattern, but not drawing the arc's outline.

Partial Arcs

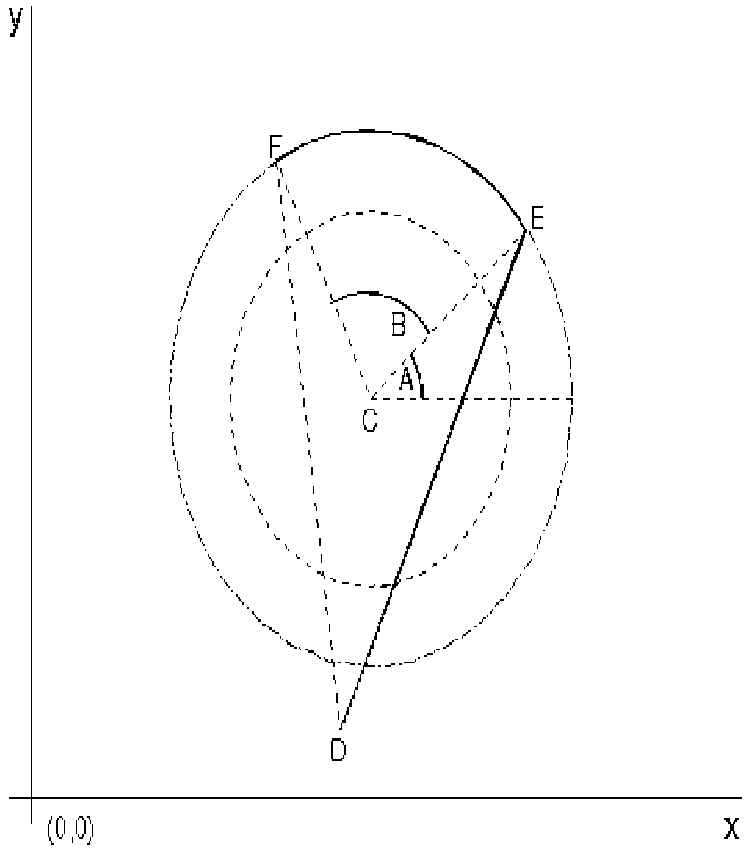
A partial arc is a section of a full arc defined by the current arc parameters. To draw a partial arc, use GpiPartialArc, which draws two separate figures. The first figure is a straight line from the current position to the starting point of a partial arc, and the second figure is the partial arc itself. When the arc has been drawn, the new current position is at the end point of the partial arc.

GpiPartialArc accepts as input the center of the current full arc, of which the partial arc is a part, specified in world coordinates. You also can specify a multiplier value to increase or decrease the size of the partial arc in relation to the current full arc.

You also must specify two positive fixed values: a start angle and a sweep angle. If the current full arc is a circle, the start angle is measured counterclockwise from the x-axis of the circle. The intersection of the start angle with the full arc, adjusted by the multiplier value, defines the starting point of the partial arc.

The sweep angle continues the counterclockwise measurement, beginning where the start angle left off. The intersection of the sweep angle with the full arc, adjusted by the multiplier value, defines the partial arc.

If the current arc is not a circle, the start and sweep angle are skewed to the same degree that the ellipse is a skewed circle. The following figure shows how the partial arc is constructed.

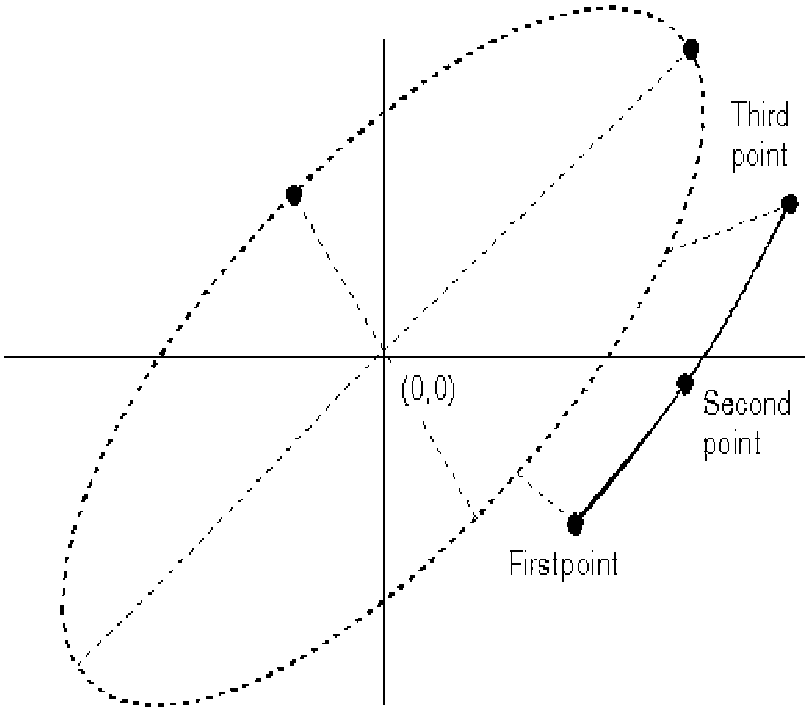


The Partial Arc

A is the start angle; B, the sweep angle. C is the center point; D, the current position. Arc.EF is the partial arc. F is the new current position. The inner circle in the previous figure is the arc defined by the current arc parameters. A multiplier has been specified with GpiPartialArc, so the partial arc is based on the full arc described by the outer circle. Point C is the center point specified on GpiPartialArc. Angle A is the start angle, and angle B is the sweep angle. GpiPartialArc therefore, draws a line from the current position (point D) to the start of the partial arc (point E). It also draws arc EF. Point F is the new current position. The arc is drawn counterclockwise because the current arc parameters define a counterclockwise circle. You can join points F and D with GpiLine. This line is drawn automatically if you define the partial arc within a GpiBeginArea and GpiEndArea bracket.

3-Point Arcs

GpiPointArc draws an arc from the current position through an intermediate point to an end point. When the arc is drawn, the current position is at the end point of the arc. You specify both the intermediate point and the end point, and these values determine both the size of the arc and the direction in which it is drawn. The shape and the orientation of a 3-point arc are determined by the current arc parameters. The following figure shows how a 3-point arc is constructed.



The 3-Point Arc

If the current arc parameters define an ellipse, that ellipse is scaled up or down to fit the three points of the arc.

When you want the three points of the arc to be points on a circle, allow the current arc parameters to default so they define a unit circle. You do not need to use the values **p** and **q** to specify the radius of the circle, because the radius is determined by the relative positions of the three points of the arc.

Multiple-Arc Primitive Family

The following table describes the three variants of the simple-arc primitive and the functions that draw them. As with line primitives, an application draws simple arcs by first using GpiMove or GpiSetCurrentPosition to set the current position. The multiple arcs are the most sophisticated of the arc primitives and their construction does not depend on the current arc parameters.

Functions that Draw Multiple Arcs

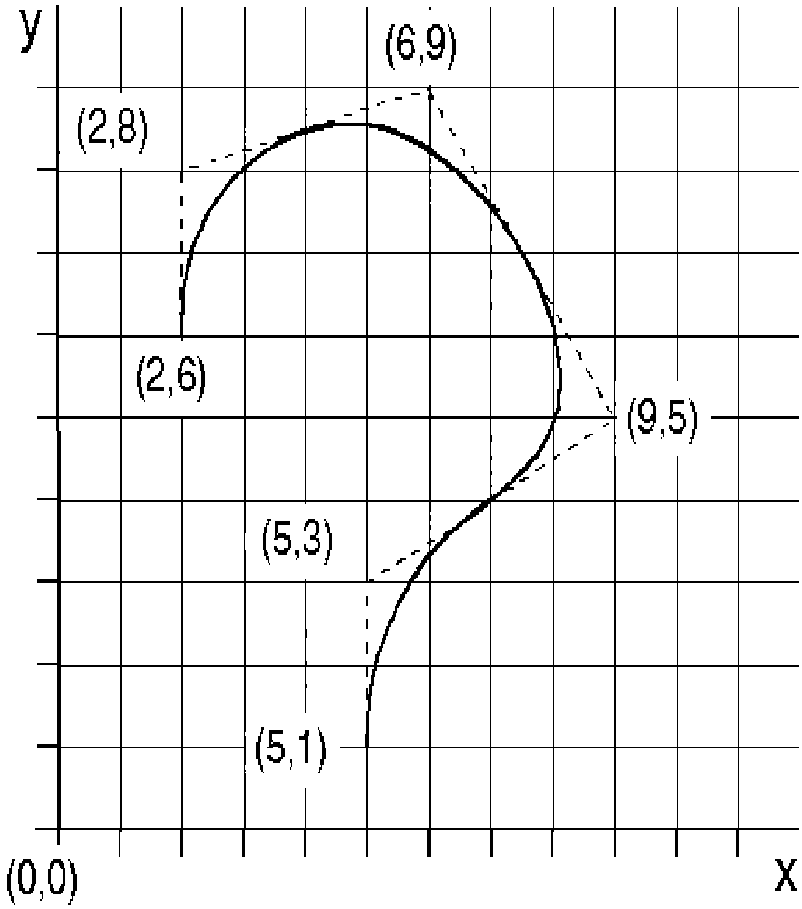
Variants	Function	Description
Fillets	GpiPolyFillet	Draws one or more fillets.
	GpiPolyFilletSharp	Draws one or more fillets with varying degrees of sharpness.
Splines	GpiPolySpline	Draws one or more splines.

Fillets

GpiPolyFillet constructs a *fillet* (a curved line) made up of one or more arcs, each of which touches a different straight line. You specify the

end points of these straight lines with GpiPolyFillet. The lines are not drawn but are used to construct the curve.

The fillet starts at the current position and finishes at the end point of the last line. On the way from the start point to the end point, the fillet is tangential to all intermediate lines at their *midpoints* . When the fillet is drawn, the current position is at the end point of the last construction line. The following figure is an example of how a fillet is constructed.



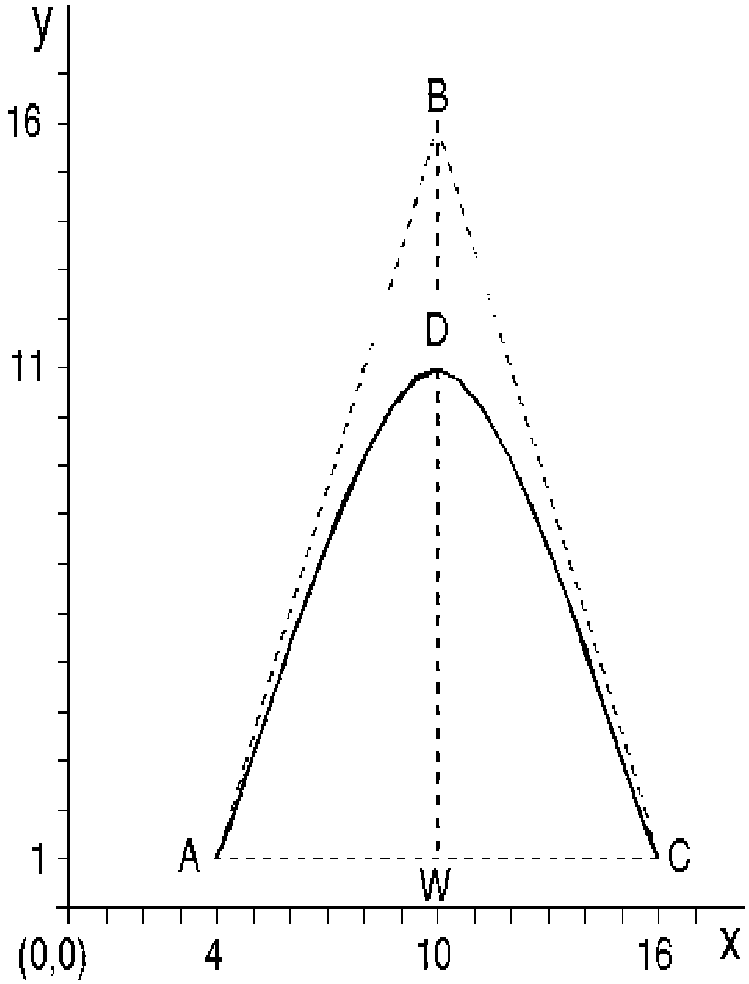
The Fillet

The fillet starts at (2,6) and ends at (5,1). The fillet is tangential to the midpoints of the lines from (2,8) to (6,9), from (6,9) to (9,5), and from (9,5) to (5,3).

When you supply only two points, the construction lines of the fillet are drawn from the current position to the first point, and from the first point to the second point. The fillet is drawn from the current position to the second point, and is tangential to the construction lines at those same points.

GpiPolyFilletSharp creates a fillet on a series of connected construction lines. The first fillet in the series is built using two construction lines: one drawn from the current position to point 1 (the control point), and one drawn from point 1 to point 2 (the end point). The fillet is drawn from the current position to the end point, and is tangential to the two construction lines at those points.

GpiPolyFilletSharp also accepts as input a sharpness value. *Sharpness* is a measure of the distance between the fillet and the control point. It is calculated as shown in the following figure.



Fillet with Sharpness Specified

Point A is the current position, point B is the control point, and point C is the end point. W is the midpoint of the notional line AC. D is the point at which the fillet crosses the notional line WB.

The sharpness of the fillet is the value WD / DB. The line WD is 10 coordinate units, and the line DB is 5 coordinate units; therefore the sharpness value is 2. The sharpness value defines the type of arc. This is shown in the following table.

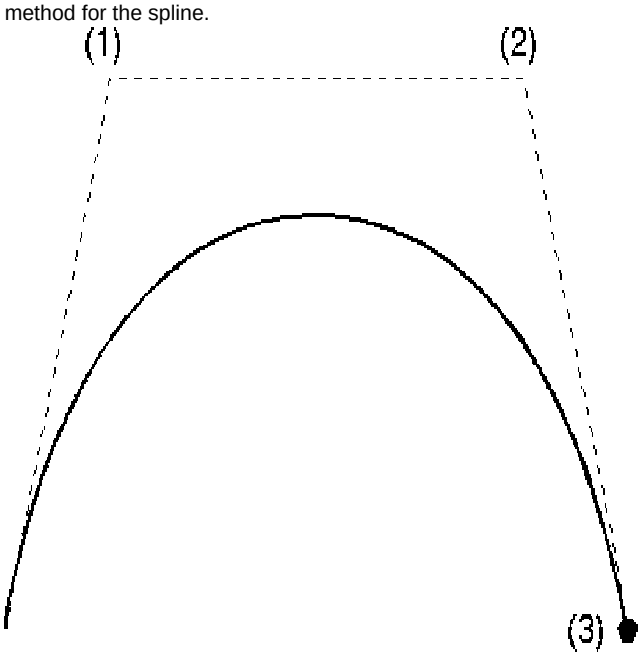
A sharpness value of...	Defines...
Greater than 1.0	A hyperbola
Equal to 1.0	A parabola
Less than 1.0	An ellipse

Subsequent fillets start from the end point of the previous fillet, and are constructed using the next two lines in the sequence in exactly the same way. For each fillet you define one control point, one end point, and one sharpness value. Upon completion, the current position is at the end point of the final construction line in the sequence.

There might be discontinuity of gradient between multiple fillets drawn with GpiPolyFilletSharp. To avoid this, ensure that points B and C of one fillet are on the same construction line as points A and B of the next fillet in the sequence. This concept is illustrated in connection with the spline primitive in the figure that follows the next figure. Discontinuity of gradient between fillets does not occur when the fillets are drawn with GpiPolyFillet.

Splines

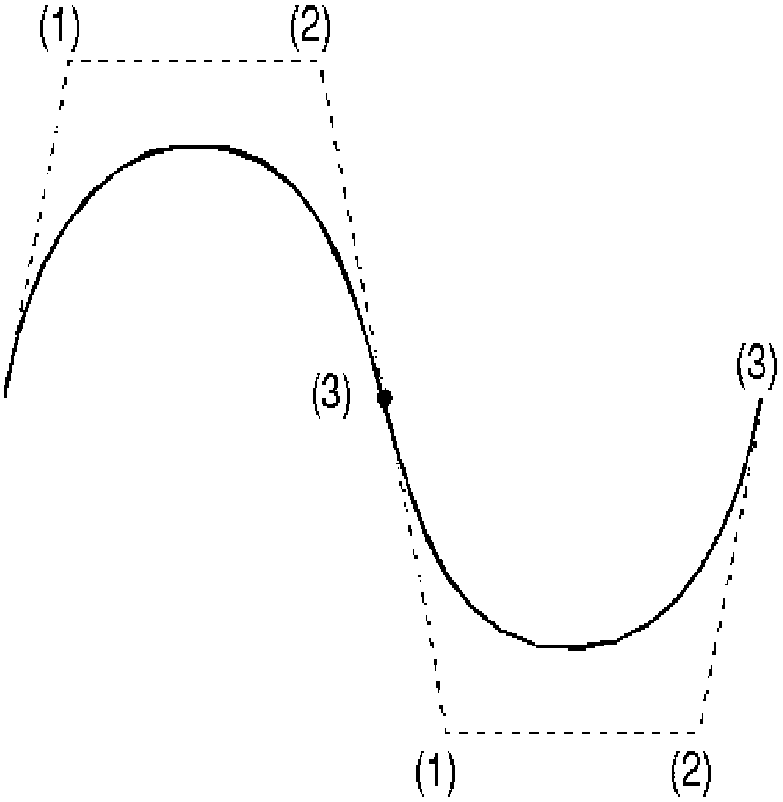
GpiPolySpline creates a succession of one or more *Bezier splines* . The spline also is a curve, but its construction method is different from that of the fillet. As input to this function, you supply three construction points for each spline. The first spline starts from the current position and ends at the third specified point. The two intermediate points are control points for the curve. Subsequent splines start at the end point of the previous spline, have two intermediary control points, and end at the third control point. The following figure shows the construction method for the spline.



The Spline

Points (1) and (2) are the control points of the spline, and point (3) is the end point.

To avoid discontinuity of gradient between the end of one spline and the start of the next, ensure that the last two construction points of the first spline and the first two construction points of the second spline are positioned along a single construction line. This concept is shown in the following figure.



Spline with No Discontinuity of Gradient

The last two points of the first spline (points 2 and 3) are positioned along the same construction line as the first two points (current position and point 1) of the second spline.

Using Line and Arc Primitives

This section explains how to:

- Draw a straight line
- Create a "rubber-banding" effect with straight lines or arcs
- Draw a circle, ellipse, fillet, or spline

Drawing a Straight Line

To draw a straight line, set the current position with GpiMove or GpiSetCurrentPosition. Set the end point of the line by filling in a POINTL structure, then draw the line with GpiLine. The following figure shows how to draw a straight line in a PM application.

```
#include <os2.h>

    BOOL DrawLine(HPS hps, LONG xStart, LONG yStart, LONG xEnd, LONG yEnd){
        POINTL ptl;                     /* Point structure              */

        ptl.x = xStart;                 /* Loads starting x-coordinate  */
```

```
        ptl.y = yStart;                  /* Loads starting y-coordinate  */
        GpiMove(hps, &ptl);              /* Sets current position        */
        ptl.x = xEnd;                    /* Loads ending x-coordinate    */
        ptl.y = yEnd;                    /* Loads ending y-coordinate    */
        if (GpiLine(hps, &ptl) == GPI_OK){
            return TRUE;                 /* Draw Line.                   */
        } /* if */
        else return FALSE;
    } /* DrawLine */
```

The second argument of GpiMove is the address of a POINTL structure that contains coordinates of the line's starting point. The second argument of GpiLine is the address of another POINTL structure that contains the coordinates of the last point on the line.

Creating a Rubber-Banding Effect with a Straight Line

When lines are drawn with a rubber-banding effect, two things happen: the original line (if one exists) is erased, and a new line is drawn in its place. This process typically takes place each time the mouse is dragged and continues until the mouse button is released. The quickest way to erase the original line (having ensured that it was drawn using mix attribute FM_XOR) is to redraw it using mix attribute FM_XOR. The following figure shows how to create this effect.

```
#define INCL_WININPUT
#define INCL_GPITRANSFORMS
#define INCL_GPIPRIMITIVES
#include <os2.h>
    HPS hps;                             /* Presentation-space handle  */
    LONG curr_color;

MRESULT EXPENTRY wpGeneric(HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2){
    static POINTL ptlStart;              /* Starting point of line     */
    static POINTL ptlNew;                /* Ending point of line       */
    static POINTL ptlPrev;               /* Previous end point of line */
    static BOOL fDraw;                   /* Line-drawing flag          */

    switch (msg) {
        case WM_BUTTON1DOWN:             /* User begins drawing        */
           GpiSetColor(hps, CLR_GREEN);
           ptlStart.x = (LONG) (LOUSHORT(mp1));
           ptlStart.y = (LONG) (HIUSHORT(mp1));
           GpiConvert(hps, CVTC_DEVICE, CVTC_WORLD, 1L,
                    &ptlStart);
           ptlPrev.x = ptlStart.x;
           ptlPrev.y = ptlStart.y;
           GpiMove(hps, &ptlStart);
           fDraw = TRUE;
           return ((MRESULT) TRUE);

        case WM_MOUSEMOVE:               /* User draws line             */
            if (fDraw) {
                ptlNew.x = (LONG) (LOUSHORT(mp1));
                ptlNew.y = (LONG) (HIUSHORT(mp1));
                GpiConvert(hps, CVTC_DEVICE, CVTC_WORLD, 1L,  &ptlNew);
                curr_color = GpiQueryColor(hps);
                GpiSetMix(hps, FM_XOR);
                if ((ptlStart.x != ptlPrev.x)
                || (ptlStart.y != ptlPrev.y)) {
                    GpiMove(hps, &ptlStart);
                    GpiLine(hps, &ptlPrev);
                } /* if */
                if ((ptlStart.x != ptlNew.x)
                || (ptlStart.y != ptlNew.y)) {
                    GpiMove(hps, &ptlStart);
                    GpiLine(hps, &ptlNew);
                    ptlPrev.x = ptlNew.x;
                    ptlPrev.y = ptlNew.y;
                } /* if */
                GpiSetMix(hps, FM_OVERPAINT);
            } /* if */
            return ((MRESULT) TRUE);

         case WM_BUTTON1UP:              /* User stops drawing          */
```

```
            fDraw = FALSE;
            return ((MRESULT) TRUE);
    } /* switch */
} /* wpGeneric */
```

Drawing a Circle

When drawing a circle, all the transformations between the world, model, page, and device spaces must maintain square units. This means that your application should select metric, English, or arbitrary page units instead of pels. On most devices, pels are rectangular, but not necessarily square. This also means that the Sx and Sy scaling factors should be equal. If the transformations maintain square units and the arc parameters are set to their default values, GpiFullArc produces a circle.

In the example shown in the following figure, if the page units are PU_LOENGLISH and the default transformations are set, a circle with a radius of 1/2 inch is drawn.

```
#define INCL_GPIPRIMITIVES
#include <os2.h>
    HPS hps;                                /* Presentation-space handle      */

MRESULT EXPENTRY wpClient (HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2){
    ARCPARAMS arcp;                         /* Structure for arc parameters   */
    POINTL ptlPos;                          /* Structure for current position */
    FIXED fxMult;                           /* Multiplier for circle          */

    arcp.lP = 1;
    arcp.lQ = 1;
    arcp.lR = 0;
    arcp.lS = 0;
    GpiSetArcParams(hps, &arcp);             /* Sets parameters to default    */
    ptlPos.x = 100;                          /* Loads x-coordinate            */
    ptlPos.y = 100;                          /* Loads y-coordinate            */
    GpiMove(hps, &ptlPos);                   /* Sets current position         */
    fxMult = MAKEFIXED(50, 0);               /* Sets multiplier               */
    GpiFullArc(hps, DRO_OUTLINE, fxMult);    /* Draws circle                  */
} /* wpClient */
```

The second argument to GpiFullArc DRO_OUTLINE, specifies that the operating system should draw only the outline of the circle-rather than filling the interior with the current fill pattern. The third argument, *fxMult*, specifies that the operating system should multiply the size of the circle by 50 units. Because the page units are PU_LOENGLISH and the default transformations are set, 50 units is equivalent to 1/2 inch.

Drawing an Ellipse

If you set the world, model, page, and device transformations so that they maintain square units, you can use the arc parameters to transform the shape of the unit circle to an ellipse, and draw this with GpiFullArc. The example in the following figure alters the arc parameter, p, by doubling its value, making the ellipse twice as wide horizontally as it is vertically.

In the example shown in the following figure, if the page units are PU_LOENGLISH and the default transformations are set, an ellipse with a 2-inch major axis (parallel to the x-axis) and a 1-inch minor axis (parallel to the y-axis) is drawn.

```
#define INCL_GPIPRIMITIVES
#include <os2.h>
    HPS hps;                            /* Presentation-space handle      */

MRESULT EXPENTRY wpClient (HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2){
    POINTL ptlPos;                      /* Structure for current position */
    FIXED fxMult;                       /* Multiplier for ellipse         */
    ARCPARAMS arcp;                     /* Structure for arc parameters   */
```

```
    arcp.lP = 2;
    arcp.lQ = 1;
    arcp.lR = 0;
    arcp.lS = 0;
    GpiSetArcParams(hps, &arcp);          /* Sets parameters to default */
    ptlPos.x = 200;                       /* Loads x-coordinate         */
    ptlPos.y = 100;                       /* Loads y-coordinate         */
    GpiMove(hps, &ptlPos);                /* Sets current position      */
    fxMult = MAKEFIXED(50, 0);            /* Sets multiplier            */
    GpiFullArc(hps, DRO_OUTLINE, fxMult); /* Draws circle               */
} /* wpClient */
```

Because the arc-parameter fields, lP and lQ, are set to 2 and 1, the operating system creates an ellipse with a major axis that is twice as long as the minor axis.

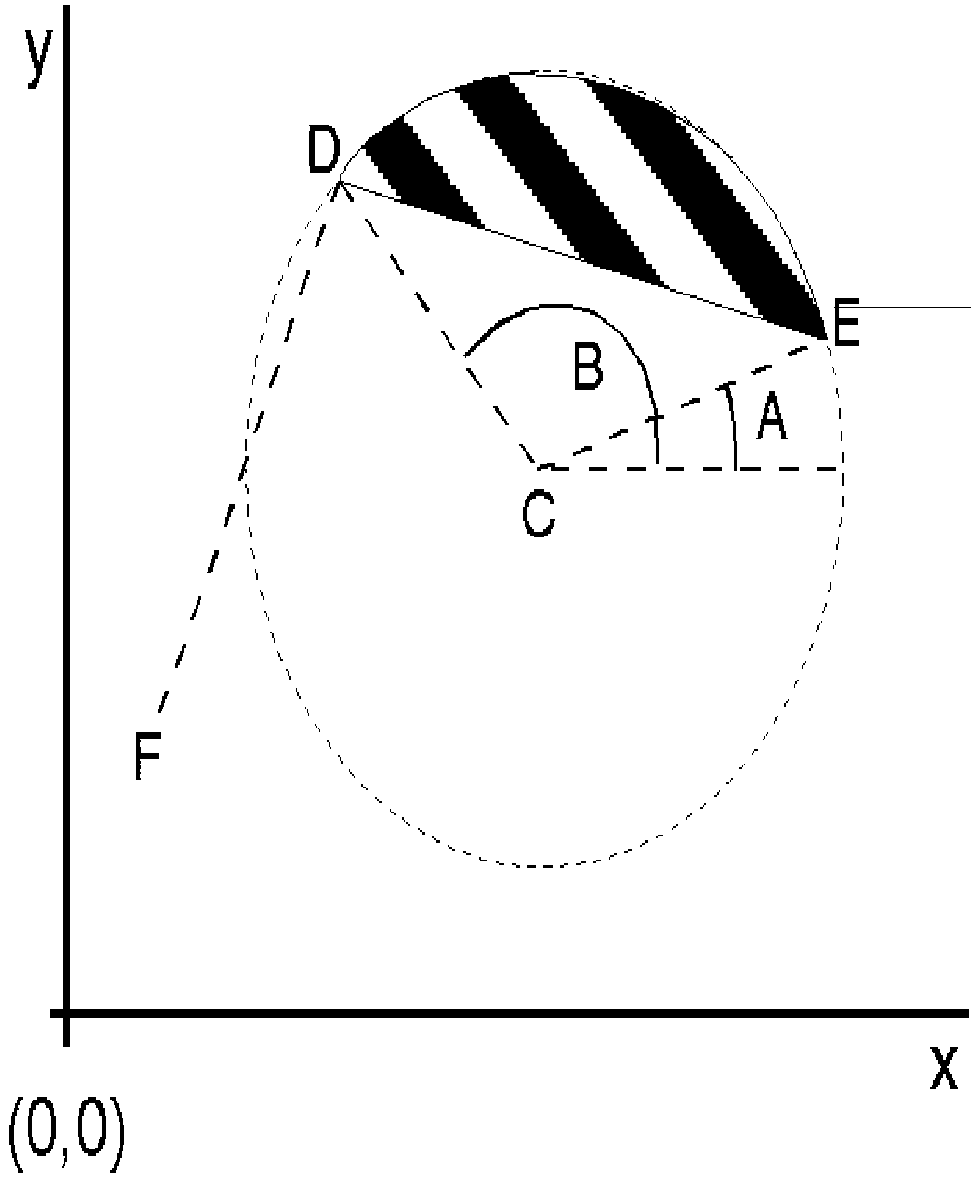
Drawing a Pie Slice

The following steps describe how to use GpiPartialArc to draw a closed shape bounded by a chord and an arc:

1. Set the current line type to LINETYPE_INVISIBLE with GpiSetLineType.
2. Call GpiPartialArc with the start angle equal to angle B, and the sweep angle equal to 0. This effectively moves the current position to a point on the current arc, and thereby defines one end of the chord.
3. Select a visible line type with GpiSetLineType
4. Call GpiPartialArc with the start angle equal to angle A, and the sweep angle equal to angle B - angle A. Angle B must be greater than angle A. The center point is the same on both GpiPartialArc calls.

To fill this partial arc with the current area-fill pattern, you can bracket the GpiPartialArc call of step 4 with GpiBeginArea and GpiEndArea. You should not call GpiBeginArea before step 2.

The effect of this sequence is shown in the following figure.



Closed Figure Bounded by Chord and Arc

The circle in the preceding figure is defined by the current arc parameters. Point F is the current position, and point C is the center of the arc as specified with the first GpiPartialArc call. The first GpiPartialArc call-with the line type set to LINETYPE_INVISIBLE-moves the current position to point D. The second GpiPartialArc call-with the line type set to LINETYPE_SOLID- draws the chord from the current position (point D) to the start point of the arc (point E), and draws arc ED.

The partial arc has been defined within an area and has been filled with the current area-fill pattern.

The following figure shows how to draw a "pie slice". like the one drawn in the previous figure.

```
#define INCL_GPIPRIMITIVES
#include <os2.h>

void Figure_516(){
   HPS hps;
   POINTL ptlCenter = {2L, 2L};       /*  Coordinates of the center point  */
   FIXED fxAngleA = 20L;              /*  Angle A in degrees               */
   FIXED fxAngleB = 130L;             /*  Angle B in degrees               */

   GpiSetLineType(hps, LINETYPE_INVISIBLE);
                  /*  Set position to start drawing the arc at angle B.  */
   GpiPartialArc(hps, &ptlCenter, MAKEFIXED(1, 0), fxAngleB, 0L);
   GpiSetLineType(hps, LINETYPE_ID);
```

```
   GpiBeginArea(hps, BA_BOUNDARY | BA_ALTERNATE);      /*  Fill the area   */
   GpiPartialArc(hps, &ptlCenter, MAKEFIXED(1, 0), fxAngleA, fxAngleB-fxAngleA);
   GpiEndArea(hps);                                    /* Cancel area-fill */
}
```

Drawing a Fillet

A fillet is tangential to two lines. The curve of the fillet is always tangential to a line drawn between its start and control points and a line drawn between its end and control points.

The following figure shows an example of how to draw a single curve, using the current position and two control points.

```
#include <os2.h>
    HPS hps;                                /* Presentation-space handle    */

MRESULT EXPENTRY wpClient(HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2){
    POINTL aptl[2];               /* Structure for control points           */

        aptl[0].x = 50;           /* Loads x-coord. of first control point  */
        aptl[0].y = 50;           /* Loads y-coord. of first control point  */
        GpiMove(hps, aptl);                /* Sets current position         */
        aptl[0].x = 75;           /* Loads x-coord. of second control point */
        aptl[0].y = 75;           /* Loads y-coord. of second control point */
        aptl[1].x = 100;          /* Loads x-coord. of third control point  */
        aptl[1].y = 50;           /* Loads y-coord. of third control point  */
        GpiPolyFillet(hps, 2, aptl); /* Draws fillet                        */
} /* wpClient */
```

When you draw a sharp fillet, the sharpness value controls the shape of the curve, as shown in a previous table.

The following figure shows an example of using a sharpness value of 3, which creates a hyperbolic curve.

```
#include <os2.h>
    HPS hps;                         /* Presentation-space handle          */

MRESULT EXPENTRY wpClient(HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2){
        POINTL aptl[2];           /* Structure for control points           */
        FIXED fxSharpness;        /* Sharpness value                        */

        aptl[0].x = 50;           /* Loads x-coord. of first control point  */
        aptl[0].y = 50;           /* Loads y-coord. of first control point  */
        GpiMove(hps, aptl);       /* Sets current position                  */
        aptl[0].x = 75;           /* Loads x-coord. of second control point */
        aptl[0].y = 75;           /* Loads y-coord. of second control point */
        aptl[1].x = 100;          /* Loads x-coord. of third control point  */
        aptl[1].y = 50;           /* Loads y-coord. of third control point  */
        fxSharpness = MAKEFIXED(3, 0);   /* Sets sharpness value            */
        GpiPolyFilletSharp(hps,   /* Draws fillet                           */
            2L, aptl, &fxSharpness);
} /* wpClient */
```

Drawing a Spline

When you use GpiPolySpline to draw a spline, each curve is tangential to the first and last of three connected lines. The following figure shows how to draw a spline.

```
#include <os2.h>
    HPS hps;                        /* Presentation-space handle             */

MRESULT EXPENTRY wpClient(HWND hwnd, ULONG msg, MPARAM mp1, MPARAM mp2){
        POINTL aptl[3];             /* Structure for control points          */

        aptl[0].x = 50;             /* Loads x-coord. of first control point  */
        aptl[0].y = 100;            /* Loads y-coord. of first control point  */
        GpiMove(hps, aptl);         /* Sets current position                  */
        aptl[0].x = 75;             /* Loads x-coord. of second control point */
        aptl[0].y = 200;            /* Loads y-coord. of second control point */
        aptl[1].x = 100;            /* Loads x-coord. of third control point  */
        aptl[1].y = 0;              /* Loads y-coord. of third control point  */
        aptl[2].x = 125;            /* Loads x-coord. of fourth control point */
        aptl[2].y = 100;            /* Loads y-coord. of fourth control point */
        GpiPolySpline(hps, 3L, aptl);   /* Draws spline                       */
} /* wpClient */
```

Marker Primitives

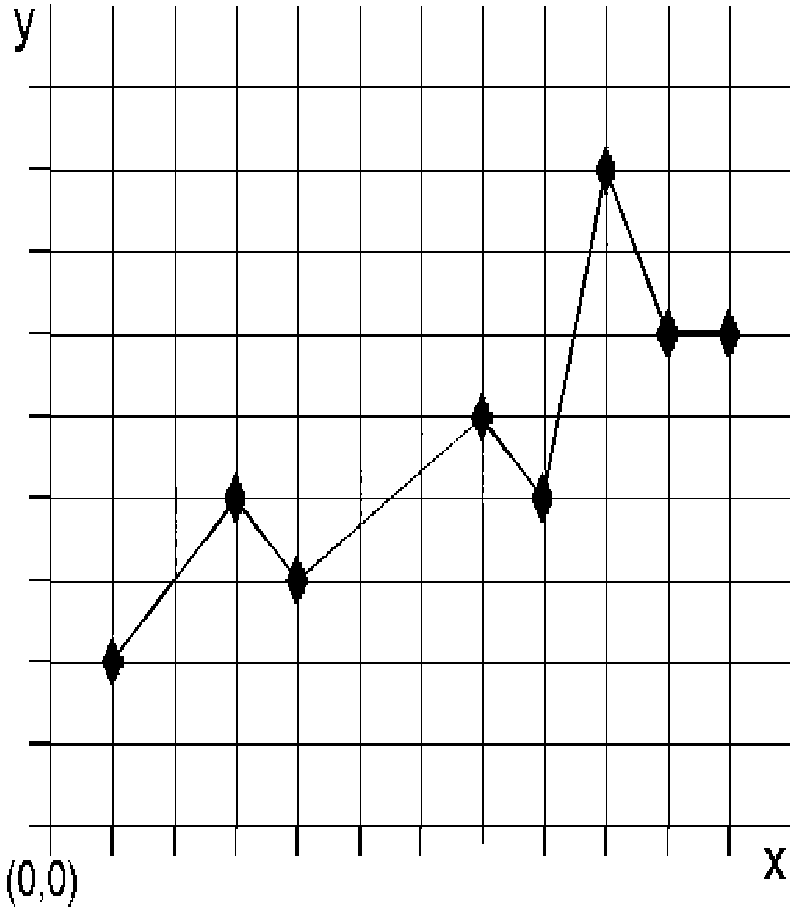
Marker primitives are graphics objects-such as stars, dots, or crosses-that are used, for example, to indicate the plotted points on a line graph. The following topics are related to the information in this chapter:

- Presentation spaces
- Line and arc primitives
- Color and mix attributes
- Fonts

About Marker Primitives

Marker primitives always are drawn centered over a point. In a designated presentation space, GpiMarker draws a single marker primitive of the current marker symbol, with its center at a specified position. This position becomes the new current position when the marker is drawn.

Another marker function, GpiPolyMarker, draws multiple marker primitives in the designated presentation space. Each marker primitive is centered over a position specified in an input array to GpiPolyMarker. All marker primitives drawn by a single call to GpiPolyMarker use the same (current) marker symbol. When a series of marker primitives is drawn, the *current position* is the center point of the last marker in the series. The following figure shows the use of marker primitives in a line graph.



Marker Primitives

This example shows a sequence of diamond-shaped marker primitives drawn on a line graph at (1,2), (3,4), (4,3), (7,5), (8,4), (9,8), (10,6), and (11,6). The new current position is at (11,6). The marker portion of this example could have been drawn with a single call to GpiPolyMarker or with eight separate GpiMarker calls.

Attributes of Marker Primitives

Marker primitive attributes are contained in a data structure called MARKERBUNDLE. Following is a list of these attributes:

- Marker symbol
- Marker box
- Marker set
- Foreground color
- Background color
- Foreground mix attribute
- Background mix attribute

When an application creates a presentation space, the marker attributes are set to the default values shown in the following table.

Marker Attribute Default Values

Attribute	Default Value	Function that Redefines Attribute
Marker symbol	Cross	GpiSetMarker
Marker box	Device dependent; equal to the size of one character	GpiSetMarkerBox

Marker set	LCID_DEFAULT	GpiSetMarkerSet Note: If this default is changed, the base marker set cannot be reselected with GpiSetMarkerSet.
Foreground color	Black	GpiSetAttrs (MBB_COLOR)
Background color	Clear	GpiSetAttrs (MBB_BACK_COLOR)
Foreground mix	Overpaint	GpiSetAttrs (MBB_MIX_MODE)
Background mix	Leave alone	GpiSetAttrs (MBB_BACK_MIX_MODE)

Marker Symbols

The current marker symbol is selected from the current marker set using GpiSetMarker. The marker symbol selected for the specified presentation space is used for all subsequent GpiMarker and GpiPolyMarker calls until a new symbol is selected.

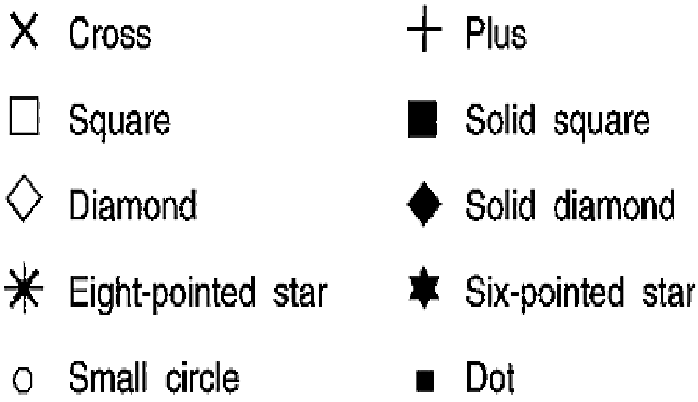
The following table describes the marker symbols provided by the PM in a base marker set. These symbols are not necessarily available from other marker sets.

The Base Marker Set

Symbol	Identifier	Long Value
Cross	MARKSYM_CROSS	1L
Plus sign	MARKSYM_PLUS	2L
Diamond	MARKSYM_DIAMOND	3L
Square	MARKSYM_SQUARE	4L
Six-point star	MARKSYM_SIXPOINTSTAR	5L
Eight-point star	MARKSYM_EIGHTPOINTSTAR	6L
Solid diamond	MARKSYM_SOLIDDIAMOND	7L
Solid square	MARKSYM_SOLIDSQUARE	8L
Dot	MARKSYM_DOT	9L
Small circle	MARKSYM_SMALLCIRCLE	10L
Blank, (Often called the invisible marker)	MARKSYM_BLANK	64L

The default marker symbol (MARKSYM_DEFAULT) is identical to the MARKSYM_CROSS symbol and has a long value of 0L. The error marker symbol (MARKSYM_ERROR) has a long value of -1L.

The following figure shows the visible marker symbols from the base marker set. Your application can determine the marker set with GpiQueryMarkerSet.



The Base Marker Set

Marker Box

The *marker box* is a rectangular boundary that defines the horizontal and vertical space occupied by the marker symbol. The marker box is used to center the current marker symbol.

If the current marker set contains *vector* marker primitives (characters outlined by using line and arc functions), changing the size of the marker box changes the size of the marker primitives also. When you change the size of the marker box, a vector marker symbol is scaled up or down automatically to fit the box. Image marker primitives, however, cannot be scaled. If the current marker set contains image characters, the marker box dimensions are fixed, because you cannot alter the size of image markers.

You can use GpiSetMarkerBox to change the size of the marker box for a particular presentation space. Set the value of the appropriate SIZEF structure to the required size. The new size is expressed in world coordinates and should not contain any fractional values.

The default size of the marker box can be determined using DevQueryCaps. The values, CAPS_MARKER_WIDTH and CAPS_MARKER_HEIGHT, depend on the currently associated device and the presentation page. Marker box values set with GpiSetMarkerBox can be determined using GpiQueryMarkerBox.

GpiSetAttrs also can be used to change the size of the marker box. GpiSetDefAttrs can be used to change the default size of the marker box.

Marker Set

When you create a presentation space, the current marker set, along with other marker attributes, are set to default. The current marker symbol (the cross described earlier) is specified from the supplied marker set. The supplied set of marker primitives contains only image, or raster, marker primitives.

The marker primitives in the default marker set are drawn by setting the color of the pels in the marker box. Within the marker box, the color of the set pels defines the foreground color. The default foreground color is neutral- black on the display screen and on printers.

The color of the pels that are not set defines the background color. The default background color is the background color on the device-white on the display screen, and the loaded paper color on printers. The default background mix is LEAVE_ALONE, or transparent, which means the background color is irrelevant.

If the default set is changed using GpiSetDefAttrs, its markers are not accessible. The markers from the default set can be recovered by calling GpiSetDefAttrs and specifying the value LCID_DEFAULT, (0).

Customizing Marker Sets

If the current marker set does not contain a symbol that suits your application, you can load a new marker set and select a marker symbol from that set. The only way to see the current marker set is to display it on the screen or view a hardcopy of the symbols. You load a new marker set for a specified presentation space by creating a logical font. Then you select the font as a marker set by specifying the logical font identifer (*lcid*) as an input parameter to GpiSetMarkerSet. After selecting the marker set, call GpiSetMarker to select a marker from the set. You can set both values simultaneously, with GpiSetAttrs.

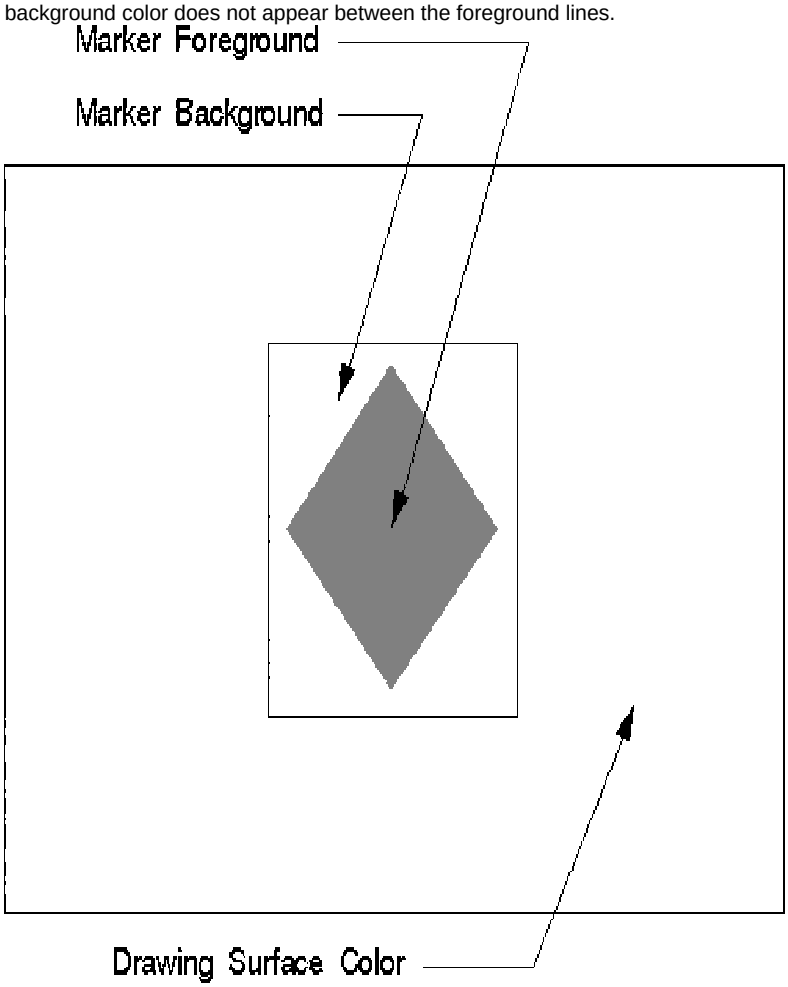
You can use any font's character set as a marker set and any character within that marker set as a marker. To determine the value that identifies the current marker set, call GpiQueryMarkerSet. To determine the value that identifies the current marker character, call GpiQueryMarker. You can retrieve both values simultaneously with GpiQueryAttrs. You also can create image marker symbols using the Font Editor.

Marker Color and Mix Attributes

The color attribute defines the color used to draw a primitive or an object. The mix attribute determines how the color of a primitive or an object is combined with the color of the drawing surface, or any other objects on the surface.

The marker color defines the color used to draw the output from any of the IBM OS/2 marker functions. When a presentation space is created, the marker color initial default is black. Markers are one of the primitives that have a foreground and background color, as shown in the following figure.

For image markers, the colors are determined by the setting of pels. For vector markers, the foreground consists of the arcs and lines that define the marker. The background color appears between the foreground lines. The marker can be solid, or filled, in which case the background color does not appear between the foreground lines.



Marker Primitives

Marker primitives have both a color and background color attribute. The mix attribute controls the combination of marker color with drawing-surface color, while the background mix attribute controls the combination of the marker box color with the drawing-surface color.

When a presentation space is created, the marker mix attribute initial default is FM_OVERPAINT. The *overpaint* mix attribute specifies that the marker color is not to be modified by the color of the drawing surface. If the marker mix attribute is changed, the marker color is mixed with colors that are already on the drawing surface.

The marker background color initial default is CLR_BACKGROUND. Usually this is defined by the application to the same color as the drawing surface. The marker background mix attribute initial default is BM_LEAVEALONE. The *leave-alone* mix background attribute specifies that the marker background color is not drawn. The box that effectively surrounds the marker appears only if the marker background mix attribute is changed.

To specify a new color or mix attribute call GpiSetAttrs. This function accepts as input the type of primitive, for example PRIM_MARKER, a list of attributes that are to be changed, a list of attributes that are to be set to their default values, and the values for the attributes that are to be changed. GpiSetAttrs is useful to specify colors and mix attributes just for a specific data structure- MARKERBUNDLE, for example. GpiSetAttrs also provides some protection against invalid colors.

To determine the current marker color and mix attribute call GpiQueryAttrs. This function accepts as input the primitive type and the attributes in question. It returns as output an array of values for the specifically queried attributes.

To reset the default marker color and mix attribute, just as with any other attribute specified in the MARKERBUNDLE data structure, call GpiSetDefAttrs. This function accepts as input the type of primitive, for example PRIM_MARKER, the attributes to be changed, and the values that will become the new default values. The changing of default values is important when working with segments. Changing the default values during a series of drawing functions is not recommended.

The marker color and mix attribute also can be specified with:

- GpiSetColor
- GpiSetMix
- GpiSetBackColor
- GpiSetBackMix

However, these functions have the disadvantage of specifying the foreground and background color or mix attribute for all primitive BUNDLE data structures that have the respective component.

There are four query functions that determine the color and mix attribute as specified by GpiSet... functions:

- GpiQueryColor
- GpiQueryMix
- GpiQueryBackColor
- GpiQueryBackMix

If the marker color, marker background color, mix attribute, or background mix attribute were specified individually, the queries can return a value inconsistent with the current marker attribute.

Using Marker Primitives

You can use marker functions to:

- Draw a single marker or a series of markers
- Set or determine (query) any combination of the marker bundle attributes including:
 - Determining the *lcid* for the current marker set
 - Selecting a character set as the new marker set
 - Determining the value that identifies the current marker
 - Selecting a character as the new marker
 - Setting or changing the size of the marker box
 - Setting or changing the color of a marker

Drawing Marker Primitives

You can draw either a single marker or a series of markers using the current marker symbol. To draw a single marker, set the fields in a POINTL structure to correspond to the desired position in world coordinates. Then call GpiMarker passing it the address of the POINTL structure as the second argument.

To draw a series of markers, set the fields in an array of POINTL structures to correspond to the desired positions in world coordinates. Then call GpiPolyMarker, passing it the number of points in the array as the second argument and the name of the array as the third argument.

The following figure shows how to draw a graph with GpiPolyLine and GpiPolyMarker.

```
#include <os2.h>
void fncMARK01(void){
    HPS hps;                                /* Presentation-space handle */
    POINTL aptl[6];                         /* Array of points           */

    aptl[0].x = 10; aptl[0].y = 15;         /* Assigns points            */
    aptl[1].x = 150; aptl[1].y = 30;
    aptl[2].x = 200; aptl[2].y = 32;
    aptl[3].x = 250; aptl[3].y = 70;
    aptl[4].x = 360; aptl[4].y = 120;
    aptl[5].x = 380; aptl[5].y = 98;
    GpiPolyMarker(hps, sizeof(aptl) / sizeof(POINTL),aptl);
                                            /* Plots points              */
    GpiMove(hps, aptl);                     /* Sets current position     */
    GpiPolyLine(hps, sizeof(aptl) / sizeof(POINTL),aptl);
                                            /* Draws lines               */
} /* fncMARK01 */
```

Selecting a New Marker

The following figure shows how to check whether the default marker primitive from the default marker set is being used currently, and if so, how to replace the cross with the six-pointed star.

```
#define INCL_GPIPRIMITIVES
#include <os2.h>
void fncMARK02(void){
    HPS hps;

    if ((GpiQueryMarker(hps) == MARKSYM_DEFAULT) &&
        (GpiQueryMarkerSet(hps) == LCID_DEFAULT))
    GpiSetMarker(hps, MARKSYM_SIXPOINTSTAR);
```

Selecting a New Marker Set

The following figure is an example of how to load a Helvetica** vector font, select it as the new marker set, and select the uppercase **A** as the new marker primitive.

```
#define INCL_GPILCIDS
#define INCL_GPIPRIMITIVES
#include <os2.h>
void fncMARK03(void){
```

```
    LONG cHelvFonts, cFonts, lcid, i, j;
    HPS hps;
    FATTRS fattrs;
    FONTMETRICS afm[80];
    MARKERBUNDLE mbnd;

    cHelvFonts = GpiQueryFonts(hps, QF_PUBLIC, "Helv",
   &cFonts, sizeof(FONTMETRICS), (PFONTMETRICS) NULL);
                                /* Queries the number of Helvetica fonts.      */

    GpiQueryFonts(hps, QF_PUBLIC, "Helv", &cHelvFonts,
       sizeof(FONTMETRICS), afm);
                                /* Loads the array of FONTMETRICS structures.  */

    for (i = 0; !(afm[i].fsDefn & FM_DEFN_OUTLINE)
                                  && i < cHelvFonts; i++);
                                /* Finds outline font                          */

    fattrs.usRecordLength = sizeof(FATTRS);
    fattrs.fsSelection = 0;
    fattrs.lMatch = afm[i].lMatch;         /* Uses Helvetica outline font       */
    for (j = 0; j <= sizeof(afm[i].szFacename); j++)
    fattrs.szFacename[j] = afm[i].szFacename[j];
    fattrs.idRegistry = 0;
    fattrs.usCodePage = 850;     /* Uses international code page               */
    fattrs.lMaxBaselineExt = 0;
    fattrs.lAveCharWidth = 0;
    fattrs.fsType = 0;
    fattrs.fsFontUse = FATTR_FONTUSE_TRANSFORMABLE;

    GpiCreateLogFont(hps, (PSTR8) NULL, lcid, &fattrs);
    mbnd.usSet = lcid;           /* Uses font as marker set                    */
    mbnd.usSymbol = 'A';         /* Uses capital A as primitive                */
    GpiSetAttrs(hps, PRIM_MARKER, MBB_SYMBOL | MBB_SET, 0, &mbnd);
} /* fncMARK03 */
```

Changing the Marker Color

The following figure shows how to set the marker foreground color to green.

```
#define INCL_GPIPRIMITIVES
#include <os2.h>
void fncMARK04(void){
    HPS hps;
    MARKERBUNDLE mbnd;

    mbnd.lColor = CLR_GREEN;
    GpiSetAttrs(hps, PRIM_MARKER, MBB_COLOR, 0L, &mbnd);
} /* fncMARK04 */
```

Matrix Multiplication

To show how matrix multiplication is implemented, here are two matrixes:

```
       a  b  c            j  k  l
```

```
d  e  f      *      m  n  o
g  h  i             p  q  r
```

The multiplication of these two matrixes produces a value for each of the nine elements of the resulting matrix:

```
element1  element2  element3
element4  element5  element6
element7  element8  element9
```

To produce element 1 of the matrix, element a is multiplied by element j, element b is multiplied by element m, and element c is multiplied by element p. That is:

*element*1 = (*a* x *j*) + (*b* x *m*) + (*c* x *p*)

To produce element 2 of the matrix, element a is multiplied by element k, element b is multiplied by element n, and element c is multiplied by element q. To produce element 3 of the matrix, elements a, b, and c are multiplied by their corresponding elements (l, o, and r) in the third vertical line of the second matrix. To produce element 4 of the matrix, you move down a row in the first matrix. That is, element d is multiplied by element j, element e is multiplied by element m, and element f is multiplied by element p. You continue the multiplication in this way until each of the nine elements has a value. The complete workings for this example are as follows:

*element*1 = (*a* x *j*) + (*b* x *m*) + (*c* x *p*)

*element*2 = (*a* x *k*) + (*b* x *n*) + (*c* x *q*)

*element*3 = (*a* x *l*) + (*b* x *o*) + (*c* x *r*)

*element*4 = (*d* x *j*) + (*e* x *m*) + (*f* x *p*)

*element*5 = (*d* x *k*) + (*e* x *n*) + (*f* x *q*)

*element*6 = (*d* x *l*) + (*e* x *o*) + (*f* x *r*)

*element*7 = (*g* x *j*) + (*h* x *m*) + (*i* x *p*)

*element*8 = (*g* x *k*) + (*h* x *n*) + (*i* x *q*)

*element*9 = (*g* x *l*) + (*h* x *o*) + (*i* x *r*)

Note that if the order of the two matrixes is reversed, the results of the multiplication are different.

Here is a simple example in which an object is scaled by a factor of 3, and then translated by (5,4):

```
3  0  0          1  0  0          3  0  0
0  3  0     *    0  1  0     =    0  3  0
0  0  1          5  4  1          5  4  1
```

You can multiply together as many transformation matrixes as you require.

Metafiles

A metafile is a graphics object that, like a segment, contains the GPI instructions that contribute to the final version of a picture. Unlike a segment, a metafile also contains a header, with state information, and all resources necessary to identically create the picture. Because pictures that are displayed when a graphics application is executed are lost when the application finishes, metafiles provide a method of

retaining pictures beyond a single execution of an application.

The following topics are related to the information in this chapter:

- Presentation spaces and device contexts
- Line primitives
- Marker primitives
- Area primitives
- Character string primitives
- Color and mix attributes
- Paths
- Regions
- Fonts
- Bit maps
- Coordinate spaces and transformations
- Print job submission and manipulation

About Metafiles

Metafiles can exist in three distinct forms. A metafile that has just been created is called a *memory metafile* because it exists in memory managed by the Presentation Manager on behalf of the application that created it. A metafile that is transferred to disk storage as a file with the default extension of .MET is called a *disk metafile* . A metafile that is loaded into an application's memory is *editable* by the application.

Metafiles save resources in the following ways:

- Metafiles can be created and used in draw mode by a single application. Therefore, an application that is not retaining segments in a segment store can retain graphics in a metafile.
- Metafiles remain available while the owning application is running, regardless of the number of presentation spaces the application obtains or defines.
- Given the same starting conditions in each presentation space, you can produce an identical picture each time the metafile contents are executed.
- Different threads or processes within an application can display a picture stored as a metafile without each having to own the metafile.
- If your organization has common graphical resources, such as a company logo, those resources can be stored in a metafile. This avoids the overhead of re-creating the resources each time they are needed.
- Because the metafile is a device-independent format, it is useful for transferring pictures that are to be printed when the printer-type is unknown.
- PM applications can exchange graphical information by using metafiles, either by using the clipboard or by transferring them over a network.
- Your application can also exchange graphics data with non-PM applications that support the Mixed Object Document Content Architecture (MO:DCA) interchange standard.

Unlike bit-maps, metafiles offer some device-independence. Bit maps store picture information on a pel-by-pel basis. Metafiles store picture information in the form of low-level graphics commands that the operating system uses to construct the pictures.

Note: Metafiles can contain bit maps or other graphics information that is in device-dependent format.

The graphics commands, called "graphics orders", represent graphics functions that create a picture. These include drawing instructions, as well as attribute-setting instructions (for example, color tables and logical fonts) and anything that describes the structure of the picture. The contents of a metafile, therefore, are similar to those of the graphics presentation space in which the picture is drawn. The Presentation Manager automatically records the environmental detail of the presentation space in which a picture is drawn in the metafile.

Note: A metafile can contain data generated from GPI functions only. Any non-graphical data included in a metafile is ignored.

An application can re-create a picture from a metafile and display it in a window on the screen or print it by "playing" the metafile. When an application displays the contents of a metafile, it can use the color table, font, fill pattern, and transformations that are stored in the metafile,

or it can use the logical color table, logical font, fill pattern, and transformations that are set for the current presentation space. The appearance of the picture stored in the metafile can, therefore, be changed by editing the current presentation space before playing the metafile.

An application can save metafiles to a disk to be loaded later by any application that chooses to access the metafile. Disk metafiles loaded into application memory can also be edited in the same manner as elements in graphics segments are edited.

They contain both primitives and attributes.

Contents of a Metafile

Every graphic function can be represented by one or more graphics orders. The operating system defines graphics orders for:

- Areas
- Bit maps (for fill patterns) and pattern-reference points
- Character output and attribute
- Colors and mix modes (foreground and background)
- Lines and arcs and line and arc attributes (including GpiBox)
- Paths (including clip paths)
- Position (for example, GpiMove)
- Saving and resetting attributes (including GpiPop)
- Transformations (in model space)
- Viewing Limits

Editing Retained Graphics and Graphics Segments also describes orders and how to edit them.

When a picture is drawn in a presentation space associated with a metafile device context, the operating system converts the primitive attributes and drawing functions into graphics orders and stores them in the metafile.

The operating system stores the effects of region and other operations that are not permitted as escape orders in the metafile. When an application finishes playing a metafile, the system restores any regions that were defined for the presentation space prior to calling GpiPlayMetaFile.

An application can use any of the query functions while creating a metafile, but the system will not store those query functions in the metafile. Similarly, an application can set a value by using a mathematical operator, but the system will store only the result of the operation in the metafile. For example, if an application determines the end point of a line by subtracting 100 from the x- and y-coordinates of the upper-right corner of a window, the system records the end point, not the relative distance, in the metafile.

Because the output of bit map functions is dependent upon the capabilities of the output device, Gpi bit map functions might produce unexpected results when they are used in a metafile. If the pel dimensions of the output device are different from the pel dimensions of the device on which the metafile was created, the bit map will be distorted. If the output device is a plotter, the bit map functions will not work at all.

Most of the escape functions used by the DevEscape function can be used in metafiles. The escape functions that the system does not save in metafiles are DEVESC_QUERYESCSUPPORT and DEVESC_GETSCALINGFACTOR.

Metafile Content Restrictions

Like areas and paths, not all functions and attributes can be represented in a metafile. Metafile content is mainly dependent upon the drawing mode in which the metafile is to be played. If the metafile is to be played when the current drawing-mode parameter is DM_DRAW (and in no other drawing mode), the metafile content is restricted as shown in Draw-Mode Restrictions. If the metafile is to be played when the current drawing-mode parameter is DM_RETAIN or DM_DRAWANDRETAIN, or if the file is to conform to SAA* guidelines, the metafile content is restricted as shown in Creating Metafiles for Interchange.

Draw-Mode Restrictions

The following restrictions apply if the metafile is to be played when the current drawing mode parameter is DM_DRAW:

- If the DCTL_DISPLAY flag of GpiSetDrawControl is OFF when you are sending graphics to a metafile, the graphics will not be visible when the metafile is played. Their effects on the current position and on current attribute settings are recorded.
- Region functions are not recorded in a metafile. The effects of the following functions, however, are recorded:

 - GpiSetClipRegion
 - GpiIntersectClipRectangle
 - GpiExcludeClipRectangle
 - GpiOffsetClipRegion
 - GpiPaintRegion

 When the metafile is played into a presentation space, temporary regions are created. Upon completion of GpiPlayMetaFile, these temporary regions are automatically deleted, and the clipping region that was current before the metafile was played is restored.
- In general, escape functions identified by reserved escape codes (that is, escape codes in the range of 0 through 32767) are recorded in a metafile. However, the DevEscape functions DEVESC_QUERYESCSUPPORT (escape code 0) and DEVESC_GETSCALINGFACTOR (escape code 1) are not stored in the metafile.
- The effect of GpiErase, including close-segment processing, is recorded in a metafile.
- The following functions can produce unexpected effects if the metafile contents are displayed on a different device from the one they were created for:

 - GpiBitBlt
 - GpiSetPel
 - GpiSetClipRegion
 - GpiOffsetClipRegion
 - GpiPaintRegion
 - GpiIntersectClipRectangle
 - GpiExcludeClipRectangle

 The reason for the unexpected effects is because the pel resolutions of the devices might differ. Raster operations (for example, GpiBitBlt) do not work on plotters.
- You can associate the metafile device context with a different presentation space while creating the metafile, but the new presentation space must have the same format as the original.

Creating Metafiles for Interchange

The following restrictions apply if the metafile is to be played when the current drawing-mode parameter is DM_DRAWANDRETAIN or DM_RETAIN, or if the metafile is to conform to SAA guidelines:

- Before you issue the first drawing instruction to a graphics presentation space that has been associated with a metafile device context, you must establish (or default) the following:

 - The graphics field (use GpiSetGraphicsField)
 - The color table (use GpiCreateLogColorTable)
 - The code page for the default character set (use GpiSetCp)
 - The default viewing transformation (use GpiSetDefaultViewMatrix)
 - The settings of the drawing controls (use GpiSetDrawControl; DCTL_DISPLAY must be set on)
 - The default values of attributes (use GpiSetDefAttrs)

- The default viewing limits (use GpiSetDefViewingLimits)
- The primitive tag (use GpiSetDefTag)
- The default arc parameters (use GpiSetDefArcParams)

You must not specify a graphics field if the metafile is to conform to SAA guidelines. If a graphics field is specified, you can play the metafile in DM_RETAIN or DM_DRAWANDRETAIN mode, but the graphics field must be specified before the first drawing instruction is issued to the metafile. The effect of the graphics field in the target presentation space when playing the metafile is controlled by the PMF_LOADTYPE option of GpiPlayMetaFile.

- You can define logical fonts and identify bit maps to be used as area-fill patterns at any time. You must not, however, issue GpiDeleteSetId against these resources after they have been established. Once assigned, therefore, *lcid*'s cannot be reused.
- The size of the logical-color-table data must not exceed 31KB.
- You must not use clipping regions. Therefore, none of the following functions are supported:
 - GpiSetClipRegion
 - GpiExcludeClipRectangle
 - GpiIntersectClipRectangle
 - GpiOffsetClipRegion
- You must not reassociate the metafile device context.
- When you use a bit map as the source of a GpiWCBitBlt operation or as an area-fill pattern, it must not be modified.
- Only the following foreground mixes can be used:
 - FM_DEFAULT
 - FM_OR
 - FM_OVERPAINT
 - FM_LEAVEALONE
- Only the following background mixes can be used:
 - BM_DEFAULT
 - BM_OVERPAINT
 - BM_LEAVEALONE
- The following functions are not supported:
 - GpiBitBlt
 - GpiSetPel
 - GpiSetPS
 - GpiResetPS
 - GpiErase
 - GpiPaintRegion
 - DevEscape

Micro Presentation Space Restrictions

When you create a metafile from a micro presentation space, or play the metafile through a micro presentation space, the contents of the metafile are restricted to those GPI functions that are valid in a micro presentation space. In this case, the application must use GpiDestroyPS instead of GpiAssociate before issuing DevCloseDC. GpiAssociate is not permitted in a micro presentation space.

Query Restrictions

Query functions can be issued, with the usual restrictions, unless the metafile device context was created using the no query option, OD_METAFILE_NOQUERY.

Segment Restrictions

Chained and unchained segments invoked by any GpiDraw... command are written to the metafile. Primitives outside segments are recorded automatically as zero segments. If the metafile is subsequently played in DM_RETAIN mode, all graphics are directed to the segment store.

Chained dynamic segments cannot be recorded in a metafile. GpiRemoveDynamics and GpiDrawDynamics cause an error condition to be raised when the presentation space is associated with a metafile device context. Unchained dynamic segments are recorded as zero segments. To draw an unchained dynamic segment, use GpiDrawSegment.

Asynchronous Drawing Thread Restrictions

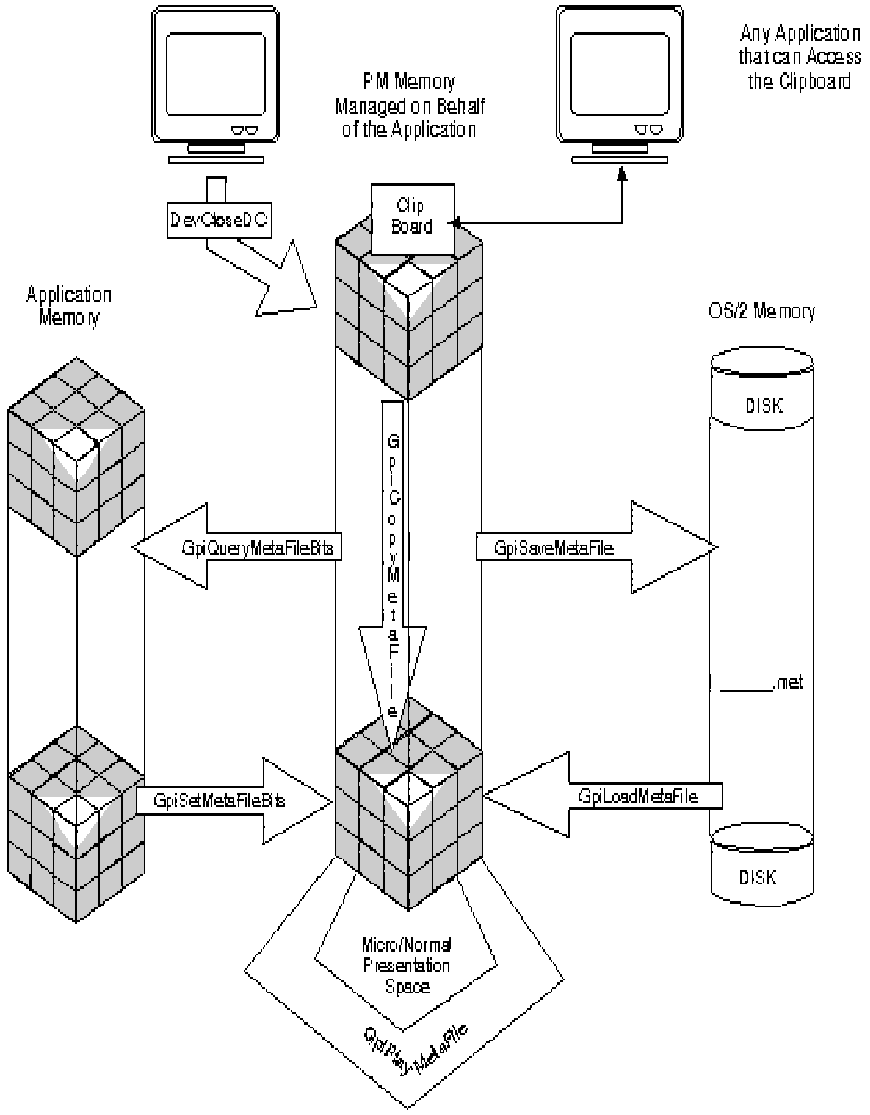
If the metafile is being created on an asynchronous drawing thread and the thread is suspended by GpiSetStopDraw, an unusable metafile results.

Metafile Functions

The OS/2 operating system provides a set of functions that allow you to:

- Create a metafile
- Store pictures in a metafile
- Play a metafile
- Save a metafile
- Load a metafile
- Edit a metafile
- Copy a metafile
- Delete a metafile

How these functions are used to create and manipulate metafiles in relationship to applications and components of the operating system is illustrated in the following figure.



Metafile Functions

Creating a Metafile

Pictures are drawn in presentation spaces associated with device contexts. PM considers a metafile to be an output device or destination, in the same manner as a window or printer. This means that to record a picture in a metafile, a metafile device context must be created and associated with a presentation space.

A metafile device context is created by calling DevOpenDC and specifying the type of device context as:

- OD_METAFILE
- OD_METAFILE_NOQUERY.

OD_METAFILE is generally used to specify a metafile device context, although OD_METAFILE_NOQUERY provides better recording performance. OD_METAFILE_NOQUERY does not support attribute queries, for example, GpiQueryCurrentPosition, or GpiQueryColor.

The device driver for the device that the metafile device context is associated with is specified in the DEVOPENSTRUC data structure required for DevOpenDC. DEVOPENSTRUC is described in Print Job Submission and Manipulation.

A metafile device context can be associated with a newly-created presentation space by calling GpiCreatePS and specifying the handle to the metafile device context returned from the call to DevOpenDC. A metafile device context can also be associated with an existing, normal presentation space by calling GpiAssociate and specifying the handle to the metafile device context returned from the call to DevOpenDC.

Storing Pictures in a Metafile

When a metafile device context is associated with a presentation space, presentation space resources (such as the current logical color table) and environmental settings (such as the presentation-page format) are copied automatically into the metafile. These items must be established *before* the presentation space is associated with the metafile device context.

Loading of additional resources (such as fonts) and adjustments to the environment (such as modifying the default viewing transform) should be made before you begin drawing. Attribute settings, segment-creation requests, and primitive-drawing requests that contribute to the picture are directed to the metafile *after* its device context has been associated with a presentation space.

If an application calls GpiSetDrawControl, specifying DCTL_DISPLAY and DCTL_OFF, before drawing graphics into a metafile, the graphics are not visible when the metafile is played. However, the metafile records any changes made to the current position or presentation-space attributes.

When the metafile device context has been associated with a graphics presentation space, the metafile is ready to receive graphical data. Just as with any other output destination, whether the picture is sent directly to the metafile is controlled by the current drawing mode, as shown in the following table.

Drawing Mode Dependencies When Recording Metafiles

Drawing Mode	Effect
Draw mode	Graphics go directly to the metafile.
Retain mode	Graphics go to the segment store of the presentation space. They are not directed to the metafile until the application issues an appropriate GpiDraw... request (GpiDrawChain, GpiDrawFrom, GpiDrawSegment).
Draw-and-retain mode	Graphics go directly to the metafile, and also to the segment store of the presentation space.

The drawing mode can be changed at any time while the metafile device context remains open by calling GpiSetDrawingMode.

As long as the metafile device context remains open, you can continue drawing. A metafile can only contain data generated from GPI functions. Any nongraphical data included in a metafile is ignored. The following list describes items found in a metafile:

- Picture
- Logical color table
- Logical font
- Fill pattern
- Viewing transformation
- Page units
- Page dimensions

When an application finishes drawing in a metafile, it must disassociate the metafile device context from the presentation space by calling GpiAssociate. If the metafile is associated with or through a micro presentation space, call GpiDestroyPS to perform an implicit disassociation.

When you have finished drawing in the metafile, and the presentation space has been disassociated, the application can close the metafile device context and obtain a handle to the metafile by calling DevCloseDC. A closed metafile cannot be reopened; therefore, additional drawing in the metafile is not possible. A closed metafile can be referenced by the *metafile handle* . The metafile handle is used to reference the metafile for subsequent operations on the metafile. Use the metafile handle to:

- Transfer a metafile to application memory
- Transfer a metafile from application memory
- Save a metafile on disk
- Play a metafile into presentation space
- Edit a metafile
- Copy a metafile
- Delete a metafile

Because each metafile can be distinctly identified, your application can work with more than one metafile at a time. However, because metafiles can be very large files, you must make maximum use of the metafile handles to avoid duplicating the actual metafiles in memory.

Playing a Metafile

An application can redraw a picture stored in a metafile by executing the contents of metafile. This is known as *playing* a metafile into a graphics presentation space. How a picture is redrawn (that is, how the metafile is replayed) depends on the current drawing-mode of the target presentation space. The following table describes these dependencies.

Drawing Mode Dependencies When Playing Metafiles

Drawing Mode	Result	If the metafile contains a segment chain...
DM_DRAW	The metafile contents are executed, and the picture is directed to the current output device.	The contents of the segments in the chain represent the picture output that is directed at the target output device.
DM_RETAIN	The metafile contents are retained in the application's segment store. The picture defined in the metafile is not directed to an output device until the application issues an appropriate GpiDraw... function.	The chain is added to the end of any chain that may already be in the segment store. If any segment in the metafile has the same nonzero name as a segment already in the segment store of the presentation space, an error condition is raised.
DM_DRAWANDRETAIN	The metafile contents are both stored and executed.	See above.

Note: Unchained segments in the metafile are always retained, regardless of the current drawing-mode parameter.

When the contents of a metafile are retained in the segment store of the target presentation space, they can be manipulated by the application as if the application had created them. For example, the segments can be edited, drawn, correlated, and deleted.

A metafile is "played" by calling GpiPlayMetaFile. A metafile cannot, however, be "played" within a segment bracket. GpiPlayMetaFile requires a metafile handle, an option count, an options array, a byte count of a descriptive record, and the descriptive record.

The GpiPlayMetaFile descriptive record is a natural-language description of the picture contents. For example, its value might be *A House*. This description is provided on input to the DevOpenDC function that defines the metafile device context. It can be used, for example, in a list box from which a user can select a picture.

The GpiPlayMetaFile options array specifies how the operating system alters your application's presentation space before playing the metafile. The options array determines the display attributes of the metafile. The array fields and their respective attributes, as numbered consecutively in the array, are shown in the following table.

Option	Attribute

PMF_SEGBASE	Reserved, must be 0.
PMF_LOADTYPE	Viewing transform, graphics.
PMF_RESOLVE	Reserved, must be 0.
PMF_LCIDS	Font and local bit map local identifiers.
PMF_RESET	Drawing attributes.
PMF_SUPPRESS	Actual playing of the metafile.
PMF_COLORTABLES	Color tables, color palettes.
PMF_COLORREALIZABLE	Realization of metafile color table.
PMF_DEFAULTS	Drawing defaults values.

PMF_RESET Option

The PMF_RESET option, more so than the other array options, controls the effects of other array options. This option allows a total reset of the presentation space and provides further control of the following presentation space environmental attributes:

- Page units (device transform)
- Page dimensions
- Default viewing transform

Warning: If the metafile page units, page dimensions and page coordinate format do not match those in the presentation space, playing the metafile will cause an error.

The PMF_RESET option can have one of the values shown in the following table.

PMF_RESET Options for GpiPlayMetaFile

Value	Description
RES_RESET	Allows the presentation space to be completely reset, just as if it were newly created, and the values of the aforementioned environmental attributes to be specified from the metafile into the presentation space. This option is equivalent to specifying GRES_ALL on GpiResetPS. If this option is used, the target presentation page is redefined so that it is the same size, and defined in the same units, as the metafile. It also ensures that the physical size of the metafile picture is preserved if the presentation page is defined in units other than pels, or arbitrary units.
RES_NORESET RES_DEFAULT	Prevents resetting of the presentation space. The existing presentation space values of the environmental attributes will be preserved, and their values in the metafile will be discarded. If this option is used, ensure that the

```
presentation-page units of the target
presentation space are the same as those
with which the metafile was generated,
before you play a metafile into the
presentation space. For example, if the
original presentation page is defined in
increments of 0.1 mm, the target
presentation page must be defined in the
same way.

To change the format of the target
presentation page, if the presentation-page
formats do not match, issue GpiSetPS, but
only if you know the metafile
presentation-page units.
```

If you are certain that the settings of the current presentation space are appropriate for the metafile, then you can play the metafile with the NO_RESET option. To reset some or all of the current values in the target presentation space, without redefining the presentation page, call:

1. GpiResetPS, with the GRES_ATTRS or GRES_SEGMENTS option
2. GpiPlayMetaFile, with the NO_RESET option.

The PMF_LCID option may be used to append or replace fonts from the metafile to the fonts in the presentation space. The PMF_COLORTABLES option may be used to append or replace color table entries in the presentation space from the metafile. Other options (PMF_LOADTYPE, PMF_DEFAULTS, and so on) can be used to selectively replace settings in the presentation space from the metafile.

Like the PMF_RESET option, other options, except for PMF_SUPPRESS, either:

- Allow the values in the presentation space to be preserved and those in the metafile to be discarded, or
- Allow the presentation space to be loaded using the values in the metafile.

If the first choice is used with RES_RESET, then the items controlled by the PMF_options will be left set to their default reset state, otherwise it will occur as described in the first choice above.

If the second choice is used with RES_RESET, then the items controlled by the PMF_options are as described above. If it is used with RES_NORESET, then the fonts and color tables in the metafile can append to the existing color tables and fonts in the presentation space. Existing presentation space color table entries and fonts, such as viewing transform will be replaced if respecified in the metafile.

Note: If your application uses the attributes from the metafile, the attributes that were specified in the presentation space before the metafile was played are overwritten.

PMF_SUPPRESS Option

The PMF_SUPPRESS option allows you to control the playing of the metafile. The PMF_SUPPRESS option can have one of the values shown in the following table.

PMF_SUPPRESS Options for GpiPlayMetaFile

Value	Description
SUP_SUPPRESS	The operating system does not draw the metafile on the device associated with the presentation space. This option is useful if you need to alter the presentation space before drawing it on an output device.
SUP_NOSUPPRESS SUP_DEFAULT	The operating system draws the metafile on the device associated with the presentation space.

You can use the RES_RESET option to reinitialize the presentation space and modify certain resources or presentation space

environmental attributes from the application before playing the remainder of the metafile by:

1. Saving the current state of the presentation space by calling GpiSavePS.

 Use this function only if you wish to restore the previous state of the presentation space after playing the metafile.

2. Calling GpiPlayMetaFile and specifying:

 - The PMF_RESET option with the value RES_RESET
 - The PMF_SUPPRESS option with the value SUP_SUPPRESS

 This causes GpiPlayMetaFile to set the presentation space to the specifications recorded in the metafile but not to play the picture in the metafile.

3. Making any required changes to the presentation space.

4. Calling GpiPlayMetaFile and specifying:

 - The PMF_RESET option with the value RES_NORESET
 - The PMF_SUPPRESS option with the value SUP_NOSUPPRESS

 This causes the picture to play.

5. Restoring the state of the presentation space after playing the metafile by calling GpiRestorePS.

PMF_LOADTYPE Option

The PMF_LOADTYPE option of GpiPlayMetaFile determines which viewing transformations and graphics fields affect the metafile picture. The PMF_LOADTYPE option can have one of the values shown in the following table.

PMF_LOADTYPE Options for GpiPlayMetaFile

Value	Description
LT_ORIGINALVIEW	Viewing transformations recorded in the metafile are used in the target presentation space. If changes to the graphics field are recorded in the metafile, the graphics field in the target presentation space is updated, and the picture is clipped to that graphics field. Changes to the default viewing transformation recorded in the metafile are used to update the default viewing transformation in the target presentation space.
LT_NOMODIFY LT_DEFAULT	Any viewing transformations recorded in the metafile are ignored. The metafile contents are drawn in accordance with the current viewing transformation in the target presentation space. (Note that, if you also specify RES_RESET, the current viewing transformation is reset to its default value.) The picture is clipped to any graphics field defined in the target presentation space, and any graphics field recorded in the metafile is ignored.

The PMF_RESET option RES_RESET changes values in the target presentation space. Therefore, the effect of the PMF_LOADTYPE

option should always be considered in conjunction with the PMF_RESET option. For example, if you specify both RES_RESET and LT_NOMODIFY, the default viewing transformation in the target presentation space is updated with that from the metafile, even though LT_NOMODIFY means that such values should be ignored.

PMF_LCIDS Option

The PMF_LCIDS option of GpiPlayMetaFile controls whether logical resources referenced by a local identifier (*lcid*) are loaded from the metafile. Logical resources include logical fonts and bit maps used as area-fill patterns. The PMF_LCIDS option can have one of the values shown in the following table.

PMF_LCIDS Options for GpiPlayMetaFile

Value	Description
LC_LOADDISC	Loads *lcid*-referenced resources from the metafile. If the *lcid*s of those resources are already in use in the target presentation space, the resources currently identified by those *lcid*s are deleted and their *lcid*s are freed before the metafile contents are loaded. The new fonts and bit maps replace the existing ones in the presentation space. (If the operating system uses a local identifier that the application has already defined, GpiPlayMetaFile deletes the existing identifier before using it for the metafile resource.)
LC_NOLOAD LC_DEFAULTS	The operating system uses the presentation space's logical font and custom fill pattern; it will ignore any logical font or custom fill pattern in the metafile. An application can use GpiSavePS and GpiRestorePS to maintain the local identifiers it has already defined.

PMF_COLORTABLE Option

The PMF_COLORTABLES option may be used to append or replace color table entries in the presentation space from the metafile. The PMF_COLORTABLES option can have one of the values shown in the following table.

PMF_COLORTABLES Options for GpiPlayMetaFile

Value	Description
CTAB_REPLACE	Replaces the logical color table in the presentation space with the color table in the metafile.
CTAB_REPLACEPALETTE	Replaces the current palette, if it exists, in the presentation space with the palette in the metafile.
CTAB_NOMODIFY	Maintains the logical color table in the

CTAB_DEFAULT	presentation space.

PMF_COLORREALIZABLE Option

The PMF_COLORREALIZABLE option may be used to select whether color table from the metafile is realized upon loading. The PMF_COLORREALIZEABLE option can have one of the values shown in the following table.

PMF_COLORREALIZABLE Options for GpiPlayMetaFile

Value	Description
CREA_DOREALIZE	Sets the realizable option when loading the color table.
CREA_NOREALIZE CREA_DEFAULT	Does not set the realizable option when loading the color table.

PMF_DEFAULTS Option

The PMF_DEFAULTS option can have one of the values shown in the following table.

PMF_DEFAULTS Options for GpiPlayMetaFile

Value	Description
DDEF_LOADDISC	Replaces the default attributes, default viewing limits, and default arc parameters in the presentation space with the values specified in the metafile.
DDEF_IGNORE DDEF_DEFAULTS	Maintains the default attributes, default viewing limits, and default arc parameters in the presentation space.

Saving a Metafile

A metafile is not a permanent object and, unless it is explicitly saved, disappears when the application that creates it ends. Metafiles are saved in disk files by calling GpiSaveMetaFile.

This function accepts as input the metafile handle and the name of the disk file in which the metafile is to be saved. As the disk file is created by this function, an error condition is raised if a file with this name already exists. When you call GpiSaveMetaFile, the memory version of the metafile is deleted. Its handle is no longer valid, until the file is reloaded from disk.

Loading a Metafile

A metafile saved on disk has to be reloaded before it can be used. Any application with access to the disk file can transfer the file from disk storage to application storage by calling GpiLoadMetaFile. GpiLoadMetaFile returns a metafile handle, which identifies the metafile after it has been loaded.

Editing a Metafile

Metafiles can be edited in a manner similar to editing graphics segments. To edit graphics orders in a metafile, an application transfers them into application storage (an array of bytes) by calling GpiQueryMetaFileBits. The number of graphics orders copied depends on the size of the array that the application supplies. The application can determine the size of the metafile (in bytes) by calling GpiQueryMetaFileLength. When the application has finished editing the graphics orders, it can copy them back into the metafile by calling GpiSetMetaFileBits.

Edited versions of metafiles can be saved by calling GpiSaveMetaFile. The disk file that contains the edited version of the metafile cannot have the same name as the file from which the metafile was originally loaded. GpiSaveMetaFile raises an error condition if the disk file already exists.

Copying a Metafile

A copy of a memory metafile can be made by calling GpiCopyMetaFile. PM makes a copy of the metafile and returns a new handle to the new metafile.

Deleting a Metafile

A memory metafile is deleted by calling GpiDeleteMetaFile. After the operation is complete, the handle to the metafile no longer points to a usable object. A disk file containing the metafile remains untouched by this operation. A disk metafile is deleted by using operating system commands that delete any other type of file.

Metafiles and the System Clipboard

A metafile that you have created or loaded can be made available to other applications at the same workstation by placing the handle to the metafile in the system clipboard. When a resource is passed to the clipboard, it becomes the property of the system, and is not deleted when the owning application ends. Any application that can access the clipboard can retrieve the handle to the metafile. Using the handle to the metafile, the application should copy the metafile by calling GpiCopyMetaFile before closing the clipboard. The application can then perform any metafile operation on the metafile.

Using Metafiles

Use metafile functions to:

- Create a metafile
- Draw into a metafile
- Load a metafile to disk
- Load a metafile from disk into an application
- Play a metafile
- Edit a metafile
- Copy a metafile
- Transfer metafile contents to application memory
- Transfer metafile contents from application memory

Creating and Drawing into a Metafile

To create a metafile, you must:

1. Create a metafile device context with DevOpenDC.
2. Create a presentation space with GpiCreatePS, and associate the presentation space with the metafile device context.
3. Draw into the metafile with various Gpi drawing functions.
4. Disassociate the metafile device context from the presentation space with GpiAssociate.
5. Close the metafile device context with DevCloseDC.

The following figure shows how to create a simple metafile that draws text within the borders of a three-color box.

```
#include <os2.h>
void fncMETA01(void){
    DEVOPENSTRUC dop;
    HDC hdcMeta;
    HPS hpsMeta;
    HMF hmf;
    HAB hab;
    SIZEL sizlPage;
    POINTL ptl;

    dop.pszLogAddress = (PSZ) NULL;
    dop.pszDriverName = "DISPLAY";

    hdcMeta = DevOpenDC(hab,
        OD_METAFILE,                /* Metafile device context            */
        "*",                        /* Ignores OS2.INI                    */
        2L,                         /* Uses first two fields              */
        (PDEVOPENDATA) &dop,        /* Device information                 */
        (HDC) NULLHANDLE);          /* Compatible device context          */

    hpsMeta = GpiCreatePS(hab,
        hdcMeta,                    /* Metafile device context            */
        &sizlPage,                  /* Page viewport                      */
        PU_PELS | GPIA_ASSOC);      /* Device units and associated context */

    /* Draw a box in a metafile. */

    GpiSetColor(hpsMeta, CLR_CYAN);

    ptl.x = 150; ptl.y = 200;
    GpiMove(hpsMeta, &ptl);
```

```
    ptl.x = 300; ptl.y = 275;
    GpiBox(hpsMeta, DRO_FILL, &ptl, 0L, 0L);

    GpiSetColor(hpsMeta, CLR_GREEN);

    ptl.x = 300; ptl.y = 200;
    GpiMove(hpsMeta, &ptl);

    ptl.x = 390; ptl.y = 275;
    GpiBox(hpsMeta, DRO_FILL, &ptl, 0L, 0L);


    GpiSetColor(hpsMeta, CLR_YELLOW);

    ptl.x = 390; ptl.y = 200;
    GpiMove(hpsMeta, &ptl);

    ptl.x = 530; ptl.y = 275;
    GpiBox(hpsMeta, DRO_FILL, &ptl, 0L, 0L);

    ptl.x = 175; ptl.y = 230;
    GpiMove(hpsMeta, &ptl);


    GpiSetColor(hpsMeta, CLR_PINK);

    GpiCharString(hpsMeta, 41,
        "METAFILE COPY METAFILE COPY METAFILE COPY");

    GpiAssociate(hpsMeta, (HDC) NULLHANDLE);
    hmf = DevCloseDC(hdcMeta);
} /* fncMETA01 */
```

Creating a Metafile

Drawing into a Metafile in Retain Mode

To draw into a metafile, set the drawing mode to the appropriate value for your application and then perform the drawing operations. The following figure shows an example of how to copy the contents of a segment into a metafile.

```
#define INCL_GPIMETAFILES
#define INCL_GPICONTROL
#include <os2.h>
void fncMETA02(void){
    HDC hdcMeta, hdc;
    POINTL ptl;
    HMF hmf;
    LONG alOpt[10];
    HPS hps;

    /*
     * Open a segment, assign it an identifier of 10,
     * and draw some text into it.
     */

    GpiSetDrawingMode(hps, DM_RETAIN);
    GpiOpenSegment(hps, 10L);
    ptl.x = 175; ptl.y = 230;
    GpiMove(hps, &ptl);
    GpiSetColor(hps, CLR_PINK);
    GpiCharString(hps, 41,
        "METAFILE COPY METAFILE COPY METAFILE COPY");
    GpiCloseSegment(hps);

    GpiAssociate(hps, NULLHANDLE);   /* Disassociates PS and screen DC */
    GpiAssociate(hps, hdcMeta);      /* Associates PS and meta DC      */
    GpiDrawSegment(hps, 10L);        /* Draws segment into metafile    */
```

```
    GpiAssociate(hps, NULLHANDLE);   /* Disassociates PS and meta DC   */
    hmf = DevCloseDC(hdcMeta);       /* Closes metafile                */

    GpiAssociate(hps, hdc);          /* Associates PS and screen DC    */
    GpiSetDrawingMode(hps, DM_DRAW); /* Sets drawing mode to DM_DRAW   */

    /*
     * Load the array of options for GpiPlayMetaFile
     * with default values.
     */

    alOpt[PMF_SEGBASE]  = 0;                  /* Reserved               */
    alOpt[PMF_LOADTYPE] = LT_DEFAULT;         /* Default transformations */
    alOpt[PMF_RESOLVE]  = 0;                  /* Reserved               */
    alOpt[PMF_LCIDS]    = LC_DEFAULT;         /* Uses default lcids     */
    alOpt[PMF_RESET]    = RES_DEFAULT;        /* Uses default           */
    alOpt[PMF_SUPPRESS] = SUP_DEFAULT;        /* Uses default           */
    alOpt[PMF_COLORTABLES] = CTAB_DEFAULT;    /* Uses default           */
    alOpt[PMF_COLORREALIZABLE] = CREA_DEFAULT; /* Uses default          */

    GpiPlayMetaFile(hps,                 /* Plays metafile onto screen   */
        hmf, 8L, alOpt, (PLONG) NULL, 0L, (PSZ) NULL);
} /* fncMETA02 */
```

Copying a Graphic Segment to a Metafile

If you want to create a simple drawing in a metafile for repeated display, you can set the drawing mode to DM_DRAW and draw directly into the metafile. The code in the first code fragment shows an example of how to do this. You can store named segments in the metafile that you are recording in DM_DRAW mode by bracketing your output primitive functions between GpiOpenSegment and GpiCloseSegment.

Copying a Metafile to Disk

You can copy a metafile to disk using GpiSaveMetaFile, and you can load the file back into your application using GpiLoadMetaFile. The following figure shows an example of how to copy a metafile to a file named META.MET, then load the same file back into the application and play it.

```
#define INCL_GPIMETAFILES
#include <os2.h>
void fncMETA03(void){
    HMF hmf;
    HAB hab;
    HPS hps;
    LONG alOpt[10];

    GpiSaveMetaFile(hmf, "meta.met");          /* Saves metafile on disk  */

    hmf = GpiLoadMetaFile(hab, "meta.met");    /* Loads metafile          */

    alOpt[PMF_SEGBASE]  = 0;                   /* Reserved                */
    alOpt[PMF_LOADTYPE] = LT_DEFAULT;          /* Default transformations */
    alOpt[PMF_RESOLVE]  = 0;                   /* Reserved                */
    alOpt[PMF_LCIDS]    = LC_DEFAULT;          /* Uses default lcids      */
    alOpt[PMF_RESET]    = RES_DEFAULT;         /* Uses default            */
    alOpt[PMF_SUPPRESS] = SUP_DEFAULT;         /* Uses default            */
    alOpt[PMF_COLORTABLES]     = CTAB_DEFAULT; /* Uses default            */
    alOpt[PMF_COLORREALIZABLE] = CREA_DEFAULT; /* Uses default            */

    GpiPlayMetaFile(hps, hmf, 8, alOpt, (PLONG) NULL, 0L, (PSZ) NULL);
} /* fncMETA03 */
```

Playing a Metafile

The following figure shows an example of how to play the metafile using the font, color table, and fill-pattern descriptions in your application's presentation space.

```
#define INCL_GPIMETAFILES
#include <os2.h>
void fncMETA04(void){
    HPS hps;
    HMF hmf;
    LONG alOpt[10];

    alOpt[PMF_SEGBASE]  = 0;               /* Reserved                        */
    alOpt[PMF_LOADTYPE] = LT_DEFAULT;      /* Viewing transformation in PS    */
    alOpt[PMF_RESOLVE]  = RS_DEFAULT;      /* Reserved                        */
    alOpt[PMF_LCIDS]    = LC_DEFAULT;      /* Font and fill pattern in PS     */
    alOpt[PMF_RESET]    = RES_DEFAULT;     /* Page units and dimensions in PS */
    alOpt[PMF_SUPPRESS] = SUP_DEFAULT;     /* Draws metafile into PS          */
    alOpt[PMF_COLORTABLES]     = CTAB_DEFAULT;    /* Color table in PS        */
    alOpt[PMF_COLORREALIZABLE] = CREA_DEFAULT;    /* Sets realizable option   */

    GpiPlayMetaFile(hps, hmf, 8, alOpt, (PLONG) NULL, 0L, (PSZ) NULL);
} /* fncMETA04 */
```

The following figure shows an example of how to play a metafile using the font, color table, and fill-pattern descriptions in the metafile.

```
#define INCL_GPIMETAFILES
#include <os2.h>
void fncMETA05(void){
    HPS hps;
    HMF hmf;
    LONG alOpt[10];

    alOpt[PMF_SEGBASE]  = 0;             /* Reserved                          */
    alOpt[PMF_LOADTYPE] = LT_DEFAULT;    /* Viewing transformation in PS      */
    alOpt[PMF_RESOLVE]  = RS_DEFAULT;    /* Reserved                          */
    alOpt[PMF_LCIDS]    = LC_LOADDISC;   /* Font and fill pattern in metafile */
    alOpt[PMF_RESET]    = RES_DEFAULT;   /* Page units and dimensions in PS   */
    alOpt[PMF_SUPPRESS] = SUP_DEFAULT;   /* Draws metafile into PS            */
    alOpt[PMF_COLORTABLES]     = CTAB_REPLACE;    /* Color table in metafile  */
    alOpt[PMF_COLORREALIZABLE] = CREA_DEFAULT;    /* Sets realizable option   */

    GpiPlayMetaFile(hps, hmf, 8, alOpt, (PLONG) NULL, 0L, (PSZ) NULL);
} /* fncMETA05 */
```

The following figure shows an example of how to play a metafile using the viewing transformation, page units, and presentation-page dimensions specified in the metafile.

```
#define INCL_GPIMETAFILES
#include <os2.h>
void fncMETA06(void){
    HPS hps;
    HMF hmf;
    LONG alOpt[10];

    alOpt[PMF_SEGBASE]  = 0;                    /* Reserved                    */
    alOpt[PMF_LOADTYPE] = LT_ORIGINALVIEW;      /* Viewing transformation      */
    alOpt[PMF_RESOLVE]  = RS_DEFAULT;           /* Reserved                    */
    alOpt[PMF_LCIDS]    = LC_DEFAULT;           /* Font and fill pattern in PS */
    alOpt[PMF_RESET]    = RES_RESET;            /* Page units/dimensions       */
    alOpt[PMF_SUPPRESS] = SUP_DEFAULT;          /* Draws metafile into PS      */
    alOpt[PMF_COLORTABLES]     = CTAB_DEFAULT;  /* Uses color table in PS      */
    alOpt[PMF_COLORREALIZABLE] = CREA_DEFAULT;  /* Sets realizable option      */
```

```
    GpiPlayMetaFile(hps, hmf, 8, alOpt, (PLONG) NULL, 0L, (PSZ) NULL);
} /* fncMETA06 */
```

Print Job Submission and Manipulation

The print subsystem of the OS/2 operating system provides a flexible, high-level interface between your application and an output device. Because most of the internal workings of the subsystem are shielded from both user and programmer, this chapter concentrates on the programming process required to produce hard copy output. Querying of printer resources, working under a specific user's environment, designing for application-specific requirements, and manipulation of print jobs also are described.

About the Print Subsystem

This section describes:

- The print subsystem components
- The print subsystem configuration
- Printing data flow

Print Subsystem Components

The print subsystem comprises the following software components of the operating system:

- Spooler
- Print subsystem user interface
- Queue drivers (queue processors)
- Printer drivers
- File system
- Kernel device drivers

Spooler

The *spooler* is the central coordinating process for the print subsystem. It gives the user flexibility in organizing and optimizing the use of the system's printers. The spooler has a global view of the system's printing resources, particularly in a server environment and therefore is able to make the best use of those resources.

- The spooler ensures that print output from two separately executing applications cannot be intermixed on the printer. As an optional optimization, the spooler can start a printing job before it is completely queued. Any successive jobs are queued normally.

- The spooler can print a job in the background while the user continues to use the application. Other single-tasking operating systems, such as DOS, require the user to wait until the print output is sent completely to the printer.

- The spooler can send jobs from PM applications across a network to a remote server, without the application's knowledge.

 Note: The file system handles print jobs from non-PM applications.

- Queues within the spooler can be used for various purposes. For example, one queue could be used for large print jobs that are printed at times when print demand is low. Another queue could be configured to print jobs using a special size paper. Print jobs on this queue would be held until the correct paper is loaded into the printer.
- The spooler can support a number of printers simultaneously. It can be configured so that jobs on a single queue can be shared among all the printers. This load balancing, which is particularly important in server environments, can be achieved without the application's knowledge.
- Jobs can be prioritized while in the queue. For example, an urgent job can be given a higher priority than ordinary jobs.

The spooler consists of one or more print queues, one for each printer object defined by the user. Jobs are created by applications and placed in a queue, waiting to print. When the previous job is completed, the next job in the queue is sent to the printer.

Print Job Formats

The print jobs held in a spooler queue are known as *spool files* . Spool files contain the following:

- Parameters submitted with the print job
- Print job data

Print job data is in one of two formats:

- **PM_Q_STD**

 Standard output data, PM_Q_STD is spooled as a PM metafile; that is, as a series of graphics orders stored in a packed binary format. PM_Q_STD print jobs are created through the GPI.

 An advantage of the PM_Q_STD format is that the files are smaller than PM_Q_RAW format files. The smaller size saves disk space for jobs in the spooler queue and reduces network traffic when transmitting the data to a network server.

 The content of PM_Q_STD jobs can be viewed using the PICVIEW application. This is achieved using the job-content menu on a print job in the printer object. The multi-page spool file can be shown in a device-independent manner so that the content of the job can be recognized easily.

 After a job is spooled and ready to print, the spooler sends the PM_Q_STD job to the queue driver. The queue driver replays the metafile, through the GPI, to the appropriate printer driver. The driver, in turn, converts the data to printer-specific commands, that is, a printer-specific format.

 There are some restrictions on the content of PM_Q_STD jobs that are related to the restrictions for PM metafiles. PM applications that cannot deal with these restrictions should enqueue print jobs using PM_Q_RAW, but there is an increase in required disk space and, possibly, network traffic.

 Print jobs sent to network servers that do not support PM are converted automatically to the PM_Q_RAW format by the system. The application still can continue specifying PM_Q_STD.

 Note: The effect of converting all PM_Q_STD print jobs to PM_Q_RAW can be turned on by the user's selecting **Printer-specific format** in a printer object settings page.

- **PM_Q_RAW**

 Raw data, PM_Q_RAW, is the actual printer command to print the job. For example, raw data created for an HP** LaserJet** printer contains Printer Command Language (PCL) commands; and raw data created for a PostScript** printer contains PostScript commands.

 The content of PM_Q_RAW jobs can be viewed with the system editor. This is achieved using the job content menu on a print job in the printer object. However, it is not always easy to recognize the content of a job. For example, PostScript is very hard to understand and get a visual idea of the actual output.

 Print jobs are created by the file system as a result of printing directly to the physical port using either INT 17 or INT 21 under DOS or the OS/2 DosOpen API. These print jobs always are queued using PM_Q_RAW. The actual queue chosen depends on the port used and the configuration of the print subsystem.

 PM applications also can use the PM_Q_RAW format, but the overall print-job creation process normally is slower because the printer driver has to perform more work to create the printer-specific format.

If the **Print while spooling** printer object setting is turned on, the user can perceive a faster response. In particular for a multi-page document, the first page starts printing as soon as the printer driver has completely finished converting the GPI to the printer-specific format.

Note: PM applications always should specify the PM_Q_STD format. The difference in disk space used can be from a factor of 2 to a factor of 50. The **Printer-specific format** and **Print while spooling** printer object settings can be used to configure optimal performance for the environment.

Although the base operating system supports the two print job formats described above, other formats can be supported by providing an appropriate queue driver.

Disabling the Spooler

The workplace user interface to control the spooler can be found in the System Setup folder. For special circumstances or applications, the spooler object can be used to disable the spooler. Generally, this is not recommended because the output from two applications can intermix, or one application can be paused until the first application has finished sending data to the printer.

Print Subsystem User Interface

The print subsystem user interface is composed of printer objects. The spooler implements each printer object by using a queue to hold the jobs. The queue is connected to a logical device that specifies configuration data about the actual physical device, for example, the port and printer drivers.

The print subsystem user interface performs the following basic functions:

- **Print job status**

 Opening a printer object folder displays an icon or detail view of the print jobs waiting in the spooler queue. Opening the settings on an individual job displays the parameters that were used when the job was queued. Some settings, such as the number of copies, can be changed while the print job is waiting in the queue.

- **Print job manipulation**

 Individual print jobs can be held in or released from the queue. *Holding* a print job means that it is not printed. Individual or all the print jobs in a queue can be deleted.

- **Queue manipulation**

 Queues also can be held, released, deleted, copied, or created using the printer object context menu.

- **Printer object configuration**

 Opening a printer object settings notebook enables a user to browse and modify the configuration of a printer object. For example, the printer driver can be changed if the user just obtained new printer.

Queue Driver

The queue driver is also called a *queue processor*. It is used to take print jobs from a queue and print the data using the printer driver. The print job data (either PM_Q_STD or PM_Q_RAW format) is passed through the GPI. For PM_Q_STD jobs, GpiPlayMetaFile is used; and for PM_Q_RAW jobs, the DevEscape DEVESC_RAWDATA is used.

Printer Driver

Printer drivers know all details of the printer they support; therefore, printer drivers are unique for each model of printer supported by the operating system. The printer driver is responsible for:

- Displaying a dialog that enables the user to inform the system how the physical printer is configured; for example, which paper sizes are installed.
- Displaying a dialog that enables the user to configure an individual print job; for example, which orientation (portrait or landscape) to use.
- Responding to application queries for available printer capabilities such as color, resolution and forms.
- Converting the GPI commands in a print job to the printer-specific language commands that will produce the expected output. The printer-specific commands are passed to the file system.

Port Drivers

Port drivers are dynamic link libraries (DLLs) that contain a set of 32-bit functions, which provide helper functions for the spooler and Workplace Shell. For each port driver DLL, there should be a physical port driver (SYS file) installed in the CONFIG.SYS file. The file type of a port driver is .PDR.

The operating system, by default, provides two port drivers; SERIAL.PDR supports COM1 - 4, and PARALLEL.PDR supports LPT1 - 3. For any other port, the supplier of the physical port software is responsible for providing a Presentation Manager port driver if it is necessary.

The functions exported from a port driver are:

- SplPdEnumPort
- SplPdGetPortIcon
- SplPdInitPort
- SplPdInstallPort
- SplPdQueryPort
- SplPdRemovePort
- SplPdSetPort
- SplPdTermPort

File System

The file system is involved in both the spooling process and the printing process. When non-PM applications create print data, the file system intercepts the data and places it on a spooler queue. After a printer driver has processed a print job, the file system sends the data to the appropriate file or device using a kernel device driver.

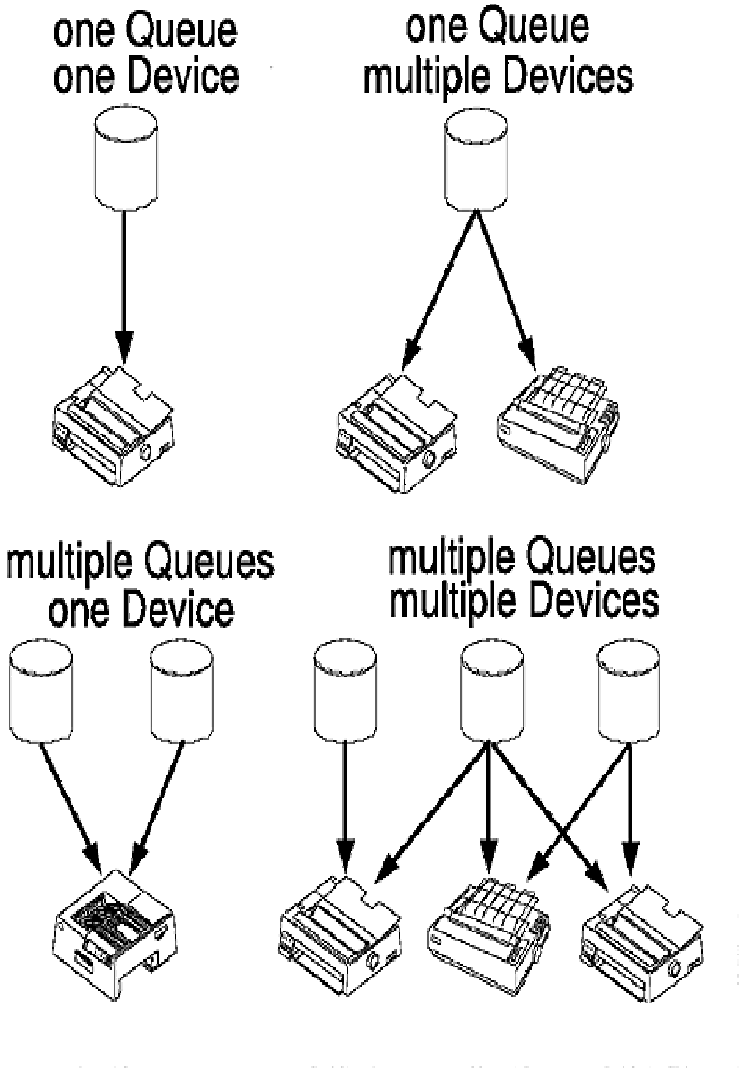
Kernel Device Driver

The system provides device drivers for physical devices. The two most commonly used by the print subsystem are the parallel device driver and the serial device driver.

Print Subsystem Configuration

From a user's viewpoint, a printer object represents a printer. The user can specify the printer object settings for configuration; for example, which printer driver and port to use.

From a programmer's viewpoint, print configuration is more complicated. Each printer object actually consists of a queue connected to a logical device. The following figure shows some example configurations. The top-left and top-right pictures show one printer object; the bottom-left picture shows two printer objects; the bottom-right picture shows three printer objects.



Example Configurations of Queues, Devices, and Printer Objects

Multiple queues connected to a single device is termed *printer sharing* . The advantage of printer sharing is that two queues can have different configurations or be used for different purposes. For example:

- One queue could be used for small jobs that are needed quickly, and the other could be used for large jobs printed during times of low demand, such as overnight.
- One queue could be used for a normal form such as a letter, and the other could be for a special form and so, would be held. When the special form was loaded into the printer, that queue could be released and the normal form queue could be held.

A single queue connected to multiple devices is termed *printer pooling* . This allows print jobs to be shared among printers for *load* balancing. The spooler does this by printing a job on the next available printer. For example, one printer usually reserved exclusively for special forms could be connected to a normal form queue during peak loads. The special form queue would be held temporarily.

Note: Configuration of a device without a queue is not allowed by the OS/2 operating system.

The logical device has the following configuration parameters that are relevant to application programmers:

- Name and description
- Logical port (for example, LPT1)
- List of printer drivers

 For example, a logical device representing an HP LaserJet printer with an installed PostScript option could have both the HP LaserJet printer driver and the PostScript printer driver in its configuration.

 Each printer driver in the list also could have printer properties associated with it. *Printer properties* describe the physical configuration of the printer, such as what size paper is installed.

The queue has the following configuration parameters that are relevant to application programmers:

- Name and description

 It is the queue *description* not the queue *name* that is used for the printer object title- that is, the text displayed beneath a printer object icon.
- Printer driver
- Default job properties

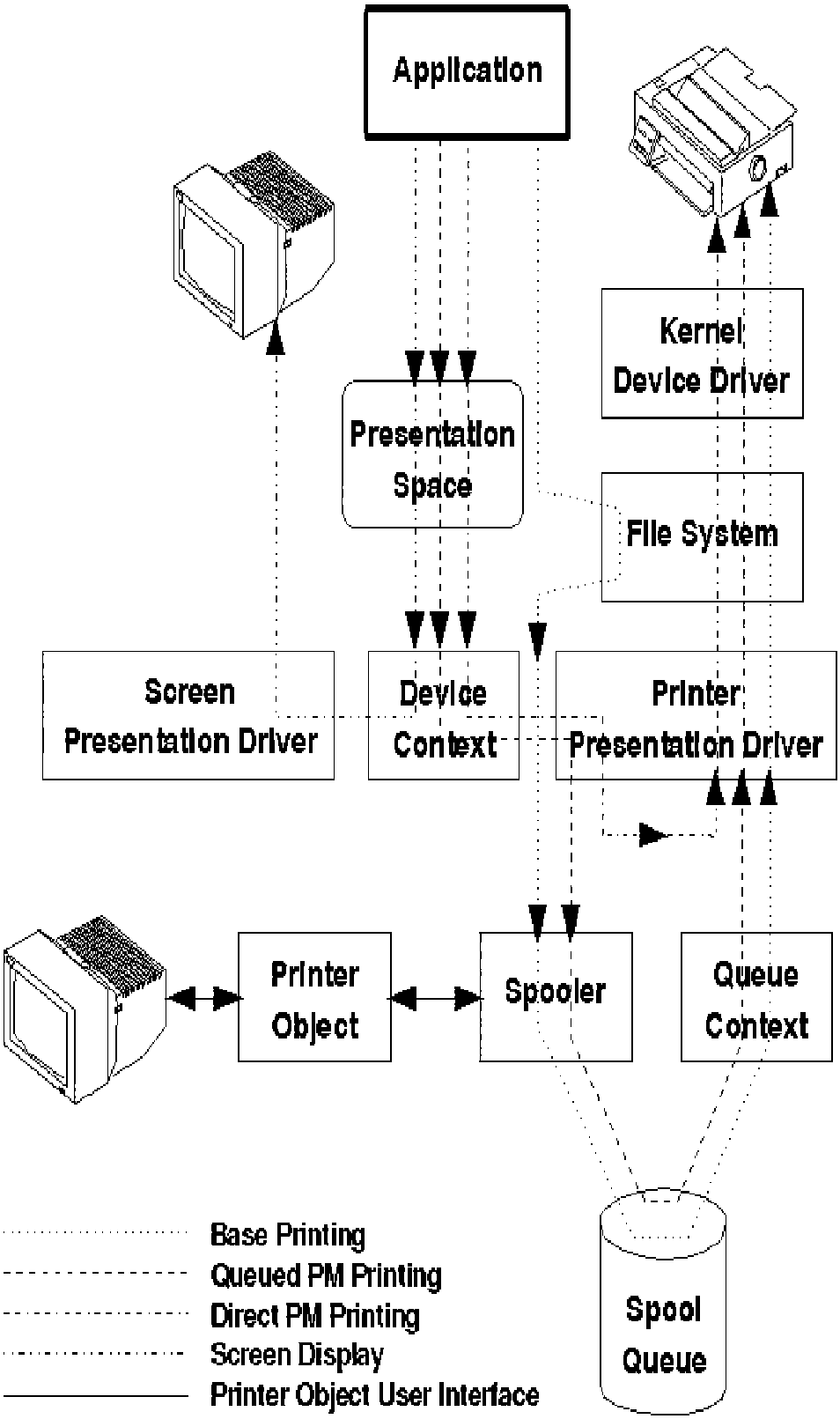
 Job properties describe the parameters that must be used for printing a specific job. An example of a job property is the orientation. The queue's job property defaults are used if none are supplied by the application.
- Queue driver
- Flags for **Printer-specific format** and **Print while spooling** settings
- Separator page
- Start and stop times

The configuration data for printer objects is stored in the OS2SYS.INI file. The spooler provides a set of functions that can be used to query and set this data.

Note: Older PM applications might still use the Profile functions (for example, PrfQueryProfileString). If so, these applications must be recoded with the new functions to avoid any dependencies on where and how the system stores configuration data.

Printing Data Flow

The data flow between your application, the printer driver, the spooler, and the kernel device driver is shown in the following figure. In addition to the data flow for the printer data stream, the data flow for screen output is shown. From an application's point of view, creating output for a printer is conceptually the same as creating screen output.



Overview of the Application Interface and Data Flow

There are three routes that the application's printer data can follow:

- **Base Printing**

Base printing is provided primarily for non-PM programs writing complete printer data streams, including all printer control codes, directly to a printer port. It also provides compatibility for applications that run under DOS or Microsoft** Windows**.

- **Queued PM Printing**

 Queued printing is recommended for all PM programs. It provides the most flexibility, both for the application and the user.

 Print jobs created by PM applications are queued on a spooler queue. The spool files are processed and finally sent to the printer asynchronously from the application.

 Note: If the spooler is disabled, queued print jobs perform as though they were submitted for PM direct printing.

- **Direct PM Printing**

 Conceptually, *direct printing* is the same as queued printing for the application interface, but the spooler is bypassed. Therefore, the data is sent directly to the printer, and the application has to wait until the printing is finished. Following are some of the reasons for PM applications to avoid direct printing:

 - The print job can interfere with other users whose jobs are destined for this printer. Print output could be mixed with the other print jobs
 - The spooler allows only one job at a time to print to a particular port to avoid mixing of print job output. If your application tries to print directly to the same port while another job is printing, your user must wait for the current job to complete before the user's job can begin.
 - The printing application loses some of its multitasking advantages because one job must complete before a second job is submitted.
 - The printer device driver does not protect a print job from job property and printer property mismatches.
 - Pictures printed directly can be different from those printed through a spooler. The default setting for direct printing is to print pictures their actual size, whereas the default for queue printing is *scaled to fit* for the output area.

 Direct printing is recommended only for specialized applications that use dedicated hard copy devices. Print jobs that have a certain degree of security associated with them, such as a corporate payroll or confidential documents, might best be handled with direct printing because files in the spooler can be copied.

 Jobs that are very large, that is, over 10MB, that you do not want copied in the spool file, also might best be printed directly.

Using the Print Subsystem

This section covers the following aspects of using the print subsystem:

- Submitting a non-Presentation Manager (base) print job
- Submitting a queued Presentation Manager print job
- Submitting a direct Presentation Manager print job
- Submitting a print job directly to the spooler
- Spooler management and configuration

Submitting a Non-Presentation Manager (Base) Print Job

A program can create a complete printer data stream including all printer control codes (called a *raw data stream*) by using DosOpen, DosWrite, and DosClose. This is called *base printing* and it is used by:

- OS/2 and DOS **Print** command
 (for example, PRINT CONFIG.SYS /D:LPT2)
- OS/2 and DOS **Copy** command
 (for example COPY CONFIG.SYS LPT1:)

- OS/2 and DOS **Redirected Output**
 (for example DIR > LPT1: or TYPE C:\CONFIG.SYS > LPT1:)
 (full screen hard copy)
- All DOS applications
- All *family* applications (applications running under DOS and OS/2)
- All Microsoft Windows applications.

Non-PM applications developed under OS/2 2.0 that must generate graphical data must be able to generate all the necessary *Escape Codes* , or printer-specific control sequences. Such applications will be coded differently, depending on the printer family-for example, Epson**, HP** LaserJet**, LaserPrinter*, Proprinter*, or PostScript**.

Non-PM applications that successfully drove the printer in the environment for which they were designed-DOS for example-will continue to do so successfully under this operating system.

DOS applications that directly access output hardware registers could have difficulty printing under the operating system. If this is so, these applications might need to be rewritten.

Submitting a Queued Presentation Manager Print Job

There are several stages involved in using the PM programming interface to print your application data. From a user interface point of view, the following four dialogs must be provided. The exact order in which these dialogs are used will govern the order of program execution.

- **Page setup dialog**

 The *page setup dialog* enables the user to specify the form name or size required. Options on this dialog can include margins on the page, header and footer strings, and whether duplex formatting is required.

 The actual contents of the dialog depend on the type of application. The important factor is that this page specify formatting options that also are used to display to a screen.

- **Printer setup dialog**

 The *printer setup dialog* displays a list of queues available to the user. A **Job properties** push button on this dialog enables the user to query and modify the job properties for this job.

- **Font dialog**

 Most PM applications must enable the user to specify the font (or fonts) required. Once the user has chosen a printer, an option on the fonts dialog enables the user to choose from device fonts as well as system fonts.

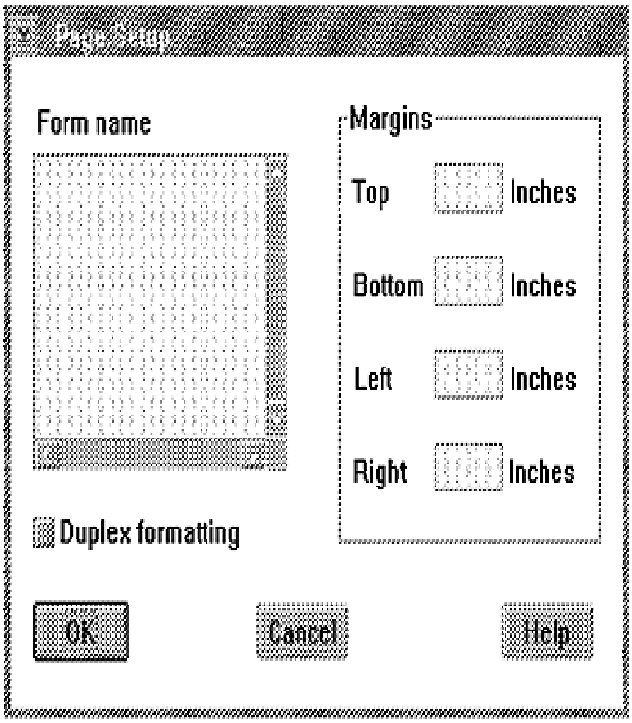
- **Print dialog**

 The application should have a Print menu item on the File menu to invoke an application-specific print dialog. It is recommended that Shift+Print Screen be used as an accelerator for printing the client area. The Print-Screen key, unshifted, is used to capture and print a window or the whole screen.

Page Setup Dialog

The page setup dialog is concerned with formatting options for the document. The user must be able to specify the form name, margins, and other application-specific formatting options such as page duplexing; that is, different formats for left and right pages in a multi-page document.

Note: The application is responsible for storing the user-defined margins for the form. The application must not allow the user to specify margins smaller than those returned by the printer driver.



Application Page Setup Dialog

Forms Selection

If the user has not chosen a specific printer, the application can supply a list of standard form sizes- for example, Letter, Legal, Ledger, A4, and A3. The application also must consider, at a minimum, a 0.4 inch (10mm) margin on the form to be within the hardware clip limits of most printers. See the following table for the common form sizes.

Form Name	Size in inches	Size in mm
Letter	8.5 x 11	216 x 279
Legal	8.5 x 14	216 x 356
Ledger	11 x 17	279 x 432
A4	8.3 x 11.7	210 x 297
A3	11.7 x 16.5	297 x 420

When a printer is chosen, it must be queried, using DevQueryHardcopyCaps for the forms it supports and the hardware clip margins. First, however, a device context must be created. It is recommended that a OD_INFO context be used, because OD_QUEUED can result in the creation of a print job. See the following figure for a sample code fragment.

```
PPRDINFO3    pprd3Device;                  /* From SplQueryDevice   */
PPRQINFO3    pprq3Queue;                   /* From SplQueryQueue    */
PSZ          pszTmp;                       /* Temporary pointer     */
HDC          hdc;                          /* Device context handle */
DEVOPENSTRUC dopData;                      /* DEVOPEN structure     */
LONG         clForms;                      /* Number of forms       */
```

```
                                        /* The device           */
PHCINFO      pchinfo;                   /* Forms information    */
                                        /* structure            */
ULONG        ulrc=DEV_OK;               /* Return code          */

/* Fill in data for devopendata for OD_INFO                      */
dopData.pszLogAddress = pprd3Device->pszLogAddr;
pszTmp = strchr(pprq3Queue->pszDriverName,'.');
if (pszTmp)
    *pszTmp = '\0';
dopData.pszDriverName = pprq3Queue->pszDriverName;
dopData.pdriv = pprq3Queue->pDriverData;

/* Open the information device context                           */

hdc = DevOpenDC ( (HAB)0,
         OD_INFO,                       /* Type                 */
         "*",                           /* Default token        */
         3L,                            /* Count                */
         &dopData,                      /* Pointer to data      */
         (HDC)0);                       /* Comp dc              */

/* Query number of forms available on the device                 */
clForms = DevQueryHardcopyCaps(hdc,
                               0L       /* Start at beginning of list */
                               0L,      /* Get number of forms        */
                               NULL);

/* Allocate memory block for forms                               */
if (DosAllocMem((PPVOID(&pchinfo), clForms*sizeof(HCINFO), fALLOC))
{
    DevCloseDC(hdc);
    return(DEV_ERROR);
}

/* Query forms data                                              */
ulrc = DevQueryHardcopyCaps(hdc,
                            0L,         /* Start with first form      */
                            clForms,    /* Query all forms            */
                            pchinfo);   /* Structure to hold returned */
                                        /* data                       */

/* Close the information device context                          */
DevCloseDC (hdc);                       /* Close the information      */
                                        /* device context             */

/* Now use forms information in pchinfo                          */
```

DevQueryHardcopyCaps returns one HCINFO structure for each form. The contents of the structure are:

- ASCIIZ name of the form (for example, Letter)
- Width and height (in millimeters) of the paper
- Clipping limits (in millimeters) of the paper
- Number of pels between clipping limits
- Whether this form is the one installed currently on the device, or whether it is selectable from another paper bin

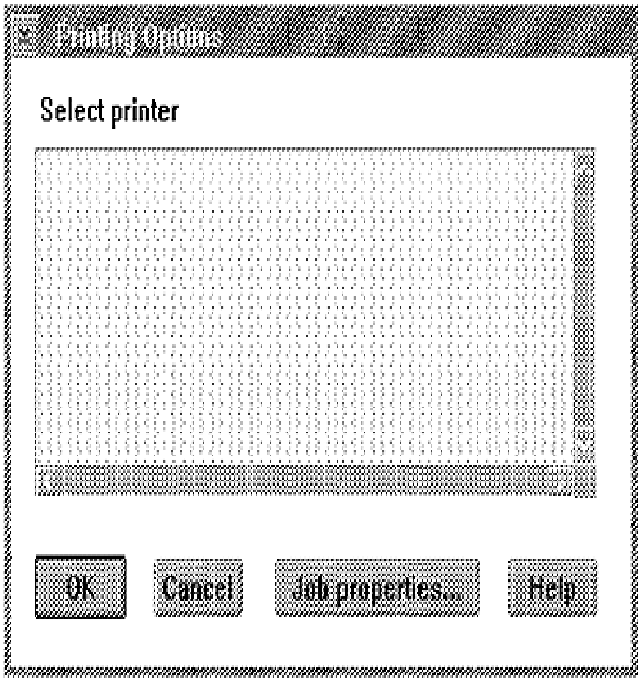
Then the list of forms can be displayed to the user. The form preselected in the list should be one of the forms marked with the HCINFO structure flag HCAPS_CURRENT.

It is recommended that an application indicate to the user which forms are currently installed in the printer. This is done by including the HCINFO structure flag HCAPS_SELECTABLE. Then users can decide whether they want a quick print on a form available from the printer or to install a different paper tray in the printer.

Printer Setup Dialog

In a printer setup dialog, an application should offer a list of printer objects that are available to the user and enable the user to select one. (The list of printer objects actually is a list of queues.) If none are available, an appropriate message must be displayed. An application must

query the list of available printer objects each time the printer setup dialog is displayed because the user might have created or modified the printer configuration while the application was executing. The following figure is an example of a printer setup dialog.



Application Printer Setup Dialog

Note: If the document was formatted for a particular device and the user selects a different printer, the application must ask the user's permission before reformatting the document for the new printer.

Use SplEnumQueue to query the list of printers (printer objects); *printer objects* essentially are spool queues. SplEnumQueue returns both a list of queues and information about each queue on the local workstation in an array of PRQINFO3 structures. It also returns information about local workstation queues that reference network print queues.

Because the number of queues might vary for each use of your application, it is essential to allocate sufficient storage to hold the data returned by SplEnumQueue. Usually the application issues the query twice: the first time, the application determines the necessary size of the information buffer; after allocating a memory block, the second query actually retrieves the information.

The SplEnumQueue parameter *pcTotal* contains the number of queues available on the local system. The application should display an appropriate message box if the value is 0.

The queue description (returned in the structure PRQINFO3 field name *pszComment*) is the printer object title. This is much more familiar to the user than the queue name, which is displayed only on the view settings page of a printer object. Therefore, the printer setup dialog should show the queue descriptions instead of the queue names.

SplEnumQueue returns information about the queue that might influence the user's or application's decision to print to this queue; for example, the queue priority, or the number of jobs already in the queue. SplEnumQueue also returns the default job properties for the queue. This data can be used by the application for the **Job properties** push button on the printer setup dialog.

Less sophisticated applications might decide to dispense with the printer setup dialog and just print to the default queue. The default queue can be queried using PrfQueryProfileString, with an application name of PM_SPOOLER and a keyname of QUEUE. The spooler function SplQueryQueue then can be called to retrieve the default job properties.

Job Properties Considerations

Job properties are options on a per-job basis- for example, orientation, resolution, and form selection. The job properties dialog is displayed

by the printer driver. Each driver has a different dialog, depending on the capabilities of the printer. The job properties are held in a printer driver-specific format in the *abGeneralData* field of the DRIVDATA structure.

Note: Job properties from one printer driver should not be given to another printer driver; the job properties probably will not be understood, and the printer driver will either return an error or substitute some default job properties. Then the user will see a change in job properties stored previously.

The user should be given the opportunity to change the job properties before the job is printed. This can be achieved by supplying a **Job properties** push button on the printer setup dialog.

Changing job properties requires two steps:

- Retrieving job properties
- Displaying the job properties dialog

Retrieving Job Properties

The application can retrieve job properties from the following:

- Previously saved job properties with the document
- Current application session job properties
- Default job properties from the queue (from **pDriverData** in the PRQINFO3 data structure)
- Printer driver device defaults (from DevPostDeviceModes using the DPDM_QUERYJOBPROP flag).

If no job properties were saved with the document, then there may be job properties that are being used by another document that also is to be printed to the same queue.

If job properties still cannot be found, then the default job properties stored with the queue can be used. It may be that the user has not set up any default job properties for the queue. Last, query the printer driver for its device defaults using DevPostDeviceModes with the DPDM_QUERYJOBPROP flag.

Displaying Job Properties Dialog

The sample code in the following figure shows how to display the job properties dialog. All the parameters required are available from the PRQINFO3 structure returned by SplEnumQueue or SplQueryQueue.

In the case of printer pooling, the *pszPrinters* field of the PRQINFO3 structure contains a list of device names, separated by commas. It is sufficient to choose the first printer in the list because the printer object ensures that the configuration of each spooled printer is the same.

Note: It is possible that the printer driver now uses a larger buffer for the job properties than the application is expecting. This occurs when a new version of a printer driver that supports some additional features is installed. In this case, the application must discard the existing document job properties, after a confirmation from the user, and query the printer driver for its device defaults, using the *DPDM_QUERYJOBPROPS* parameter to DevPostDeviceModes.

```
#define INCL_DEV
#define INCL_DOS
#include <os2.h>
#include <memory.h>

{
    ULONG         ulrc=FALSE;
    HAB           hab;
    PPRQINFO3     pprq3Queue;    /* From SplEnumQueue or SplQueryQueue */
    PSZ           pszDriverName,pszDeviceName,pszTmp;
    HDC           hdc=NULL;
```

```
    LONG          cbBuf;

/* Use the first device name in the PRQINFO3 structure                */
   pszTmp = strchr(pprq3Queue->pszPrinters, ',');
   if (pszTmp)
       *pszTmp = '\0';

/* Use just the driver name from the driver.device string             */
   pszDeviceName = strchr(pprq3Queue->pszDriverName, '.');
   if (pszDeviceName)
   {
       *pszDeviceName = '\0';
       pszDeviceName++;
   }

/* Check size of buffer required for job properties                   */
    cbBuf = DevPostDeviceModes( hab,
                                (PDRIVDATA)NULL,
                                pprq3Queue->pszDriverName,
                                pszDeviceName,
                                pprq3Queue->pszPrinters,
                                DPDM_POSTJOBPROP
                              );

/* Return error to caller                                             */
    if (cbBuf<=0)
        return(cbBuf);

/* Return BUFFER TOO SMALL error to caller                            */
    if (cbBuf > pprq3Queue->pDriverData->cb)
        return(DPDM_ERROR);

/* Display job properties dialog & get updated job properties from driver */
    ulrc = DevPostDeviceModes( hab,
                               pprq3Queue->pDriverData,
                               pprq3Queue->pszDriverName,
                               pszDeviceName,
                               pprq3Queue->pszPrinters,
                               DPDM_POSTJOBPROP
                             );
    return(ulrc);
}
```

Font Dialog and Device Font Considerations

There are two types of fonts:

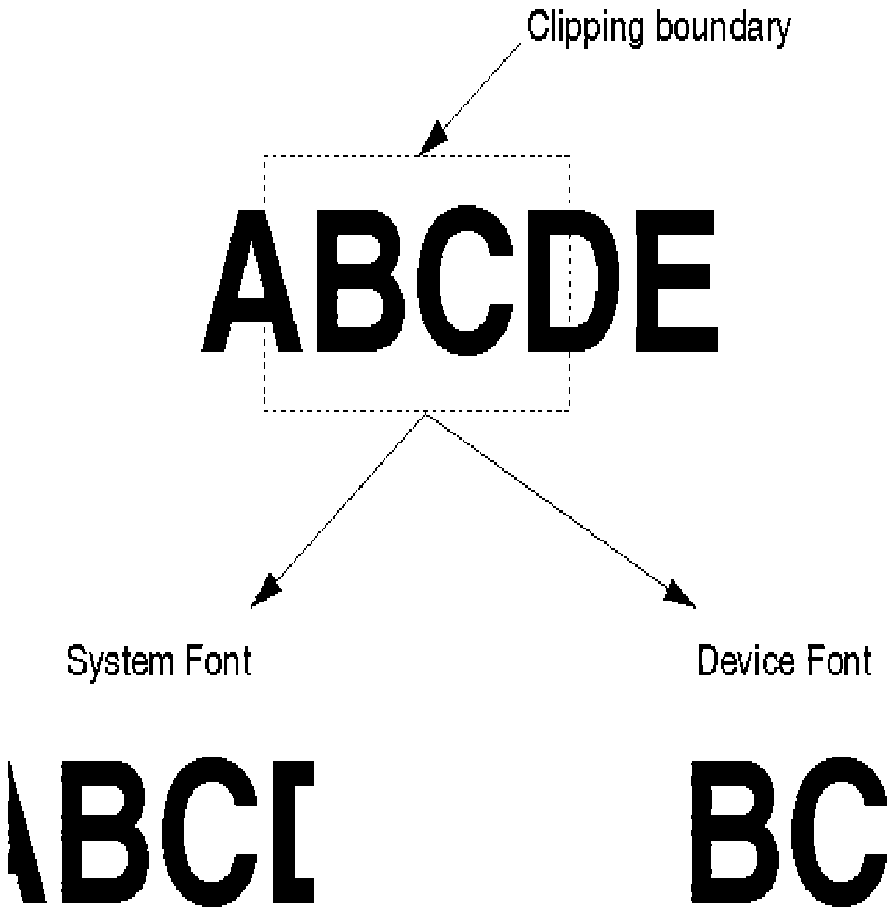
- *System fonts* , which can be used for both displays and printers, and therefore, are generally more flexible.
- *Device fonts* , which are specific to a particular device. Device fonts for printers can be built-in or be installed with font cartridges by a user. Device fonts also can be available on diskette (also termed *soft fonts*) and can be downloaded to the printer as required by the printer driver.

 The advantage of device fonts is that, usually, they are printed in the highest resolution of the device and are faster than system fonts.

Note: Some printer drivers, in particular, the HP LaserJet (LASERJET) and the LaserPrinter (IBM4019) printer drivers provide an intermediate solution. An option on the job properties dialog enables system fonts to be downloaded to the printer as soft fonts.

As stated earlier, an application must provide the user with the ability to choose fonts and, in particular, to choose a device font over a system font to achieve better performance. A standard font dialog box should be used (see WinFontDlg).

When an application uses device fonts, and the output mixes text and graphics pictures, the device fonts are clipped per character. System fonts give more precise clipping to the pel. The following figure illustrates the results from clipping two lines of text: one generated in a system font; the other, in a device font.



Character Clipping Results for Device and System Fonts

The available device fonts can be queried, using GpiQueryFonts, in either an OD_INFO or an OD_QUEUED device context. Device fonts are returned with negative *lMatch* numbers. Then the appropriate logical font can be used after calling GpiCreateLogFont.

If, during the printing process, a logical font is not created, the printer driver uses its default font for the printer, which usually is 12-point Courier.

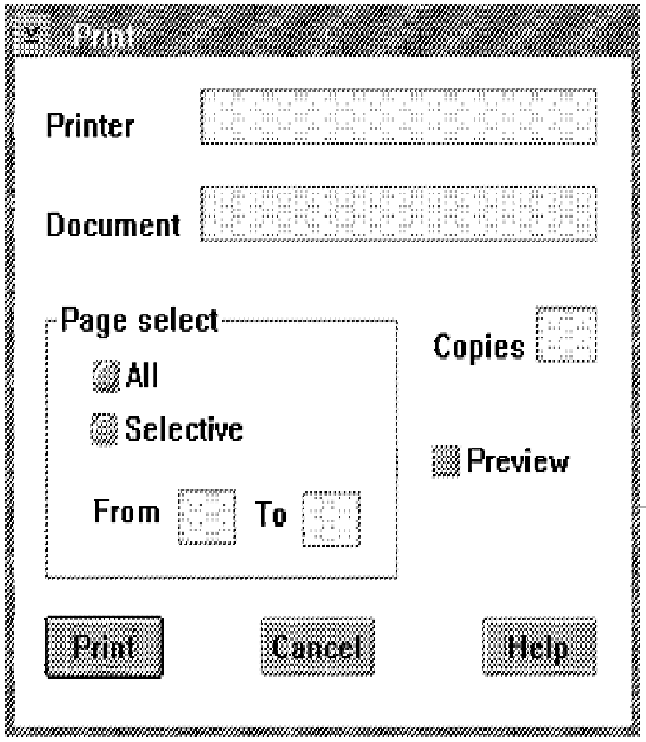
There are two design choices for device fonts:

- Use system fonts for the display. When printing, query the printer driver and attempt to choose a device font that closely matches the system font. If no match is found, then print using the system font. A device's font metrics and character spacing might not match exactly, so the printed output might not be exactly the same as the display.
- Use a printer font if the user selects one. Determine the metrics and find a system font that shares the same characteristics (for example, a style such as Courier or Serifed). The character string to be displayed is sent to the printer driver and the inter-character spacing is returned, using GpiQueryCharStringPos. The individual character positions are used when the string is sent to the display, using GpiCharStringPos with the *flOptions* parameter of CHS_VECTOR. While the display output can take longer to display to the screen, true *WYSIWYG* (*What You See Is What You Get*) is achievable.

Print Dialog and Print Processing

The print menu must display a dialog to enable the user to specify the number of copies, start page, end page, or any other

application-specific print options. A confirm push button (for example, **Print**) initiates the print. The following figure is an example of a print dialog.



Application Printer Setup Dialog

The **Printer** entry field is the read-only name of the printer object (queue) that the user chose from the Printer Setup dialog. The **Preview** checkbox is used to preview the printed output on the screen. This can be achieved by opening an OD_MEMORY device context to the printer driver and drawing the picture into the bit map. A BITBLT (Bit Block Logical Transfer) to the screen shows the resultant page. System fonts should be used instead of device fonts.

Once the user has initiated the print, a PM application must execute the following steps.

1. Create a new thread.

Then, on this new thread:

2. Open a device context.
3. Associate a presentation space.
4. Start the print job.
5. Query the device capabilities.
6. Format the page.
7. Start the next page.

Repeat steps 6 through 7 for each page.

8. End the print job.

For another print job, repeat from step 4.

9. Disassociate the presentation space.
10. Close the device context.

The following sections describe each of the above steps in turn.

Creating a New Thread

Once the print process has been confirmed by the user. the application must use a separate thread to perform the actual printing. This maintains user responsiveness and avoids displaying the hourglass pointer. While the printing is in progress, the application should dim certain menu options, such as drawing. Other options, such as page down, zoom, or help still must be available.

Opening a Queued Device Context

Before you can send data to a printer using device functions, you must open a device context with DevOpenDC. The parameters of DevOpenDC are influenced by the queue and job properties the user has chosen previously.

DevOpenDC accepts the following parameters as input:

hab	The anchor-block handle from a WinInitialize call.
lType	The type of device context. This always is OD_QUEUED to specify a queued device context.
pszToken	The device-information token. Always use an asterisk (*) for this parameter to force the system to get device information from the *pdopData* parameter.
lCount	The number of data elements supplied in the *pdopData* parameter. The minimum number of parameters for a queued device context is 4.
pdopData	The device-context data area. This is a pointer to a structure of the type DEVOPENSTRUC. The elements of this structure are described below.
hdcComp	The compatible-device-context handle. This must be NULL for a device context of type OD_QUEUED.

DevOpenDC returns a device-context handle of type HDC. The handle is used in other functions beginning with the Dev prefix and for GpiCreatePS and GpiAssociate.

The DEVOPENSTRUC structure contains all the data needed to define a device context.

The individual elements of the DEVOPENSTRUC structure are described below. At least the first four structure members must be provided for queued device contexts.

pszLogAddress	Name of the queue. It is the *pszName* field of the PRQINFO3 structure. **Note:** For device contexts of type OD_INFO, **pszLogAddress** is the port name. This can be retrieved by calling SplQueryDevice using the **pszPrinters** device name. The *pszLogAddr* field in the PRQINFO3 structure is the port name.
pszDriverName	Character string identifying the printer driver, for example, LASERJET. The *pszDriverName* field of the PRQINFO3 structure, associated with the required print queue, gives the driver and device name, separated by a period, for example LASERJET.HP LaserJet IIID The *pszDriverName* field can contain only the name up to the period, for example LASERJET.
pdriv	This is a pointer to the job properties data returned by the printer driver from DevPostDeviceModes or the default job properties from *pDriverData* in the PRQINFO3 structure. **Note:** The DRIVDATA structure contains the particular device name to be used. Therefore, it is a programming error to set this parameter to NULL.
pszDataType	It is recommended that PM_Q_STD always be used for the data type.
pszComment	Optional character string that the printer object displays to the user in a job settings notebook.

It is recommended that the application include its own name in this comment string.

Note: The job title text is derived from the document name.

pszQueueProcName

Queue processor name (optional). The queue processor (also termed *queue driver*) name is available from the *pszPrProc* field of the PRQINFO3 structure. The default queue processor provided by the operating system is PMPRINT. The user also can install a queue processor (PMPLOT) that is used to provide reverse clipping for vector devices such as plotters.

For specialized applications, it is possible to use an alternative queue processor to the default specified for the queue. The list of installed queue processors is available from the OS2SYS.INI file using the application name PM_SPOOLER_QP.

pszQueueProcParams

Queue processor parameters (optional). They can include information such as the number of copies you want to print and the size of the output area on the printed page. See PMPRINT/PMPLOT Queue Processor Parameters for details.

pszSpoolerParams

Spooler parameters (optional) are separated by spaces. They are used for scheduling print jobs and are as follows:

- The form names that identify the paper to be used, for example, **FORM=A4,A5,ENV**. The form names are optional, but if they are provided, the spooler is able to hold off printing the jobs until the required form is installed in the printer. If the form name is not provided, the spooler attempts to print the job. The printer driver recognizes that there is a forms problem and displays a FORMS MISMATCH message box.
- Priority of the print job, for example, **PRTY=60**. The priority is specified as an integer in the range 1 through 99; 99 is the highest. The default priority value is 50. The application can use the spooler priority parameter to prioritize its own jobs. However, it is not good practice for an application always to use priority 99 in an attempt to get its jobs printed first.

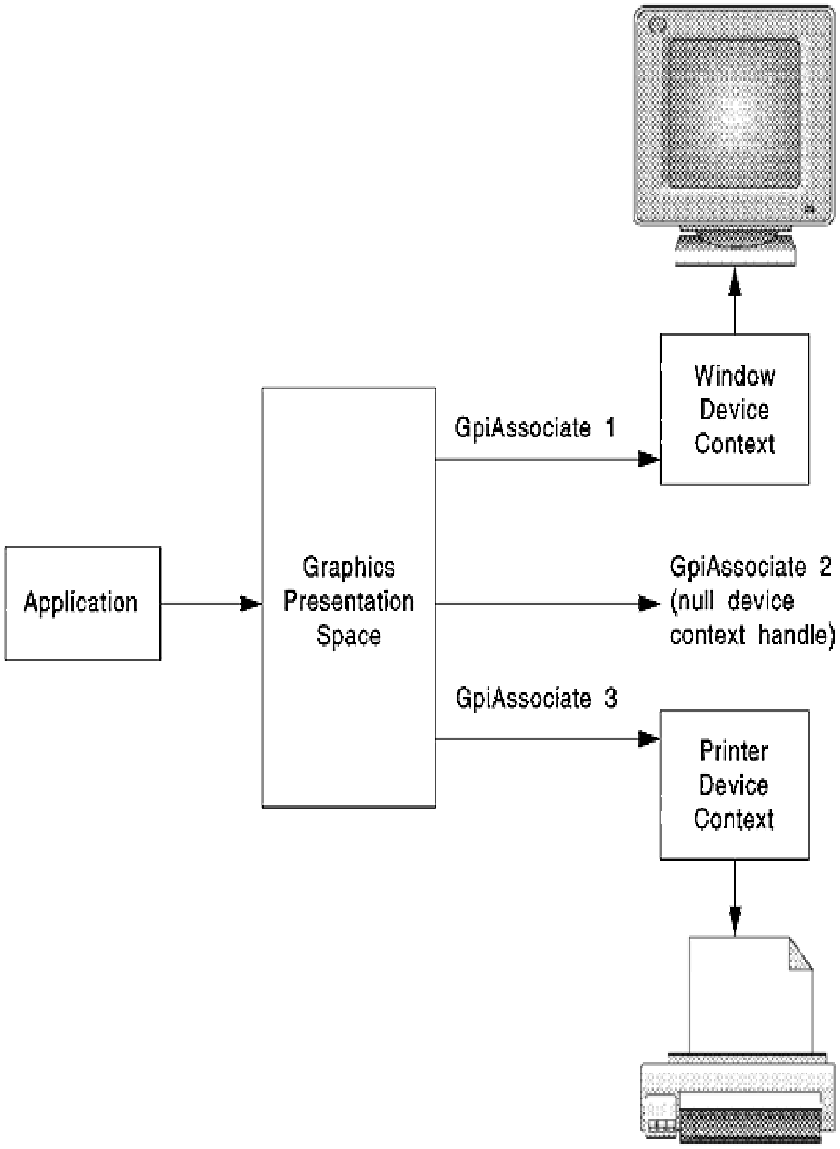
pszNetworkParams

Optional parameter that can be used to specify network options; for example, **USER=JOESMITH**.

Associating a Presentation Space

In order to print, a device context must be associated with a GPI presentation space (PS).

In most circumstances, a presentation space already exists for the screen. In such cases, the PS can be disassociated with the window device context and associated with the printer device context by using GpiAssociate, as shown in the following figure.



Reassociating Presentation Space with Device Contexts

If a presentation space does not exist or a different one is to be used, the application must call GpiCreatePS and use the GPI_ASSOC flag to associate the printer device context.

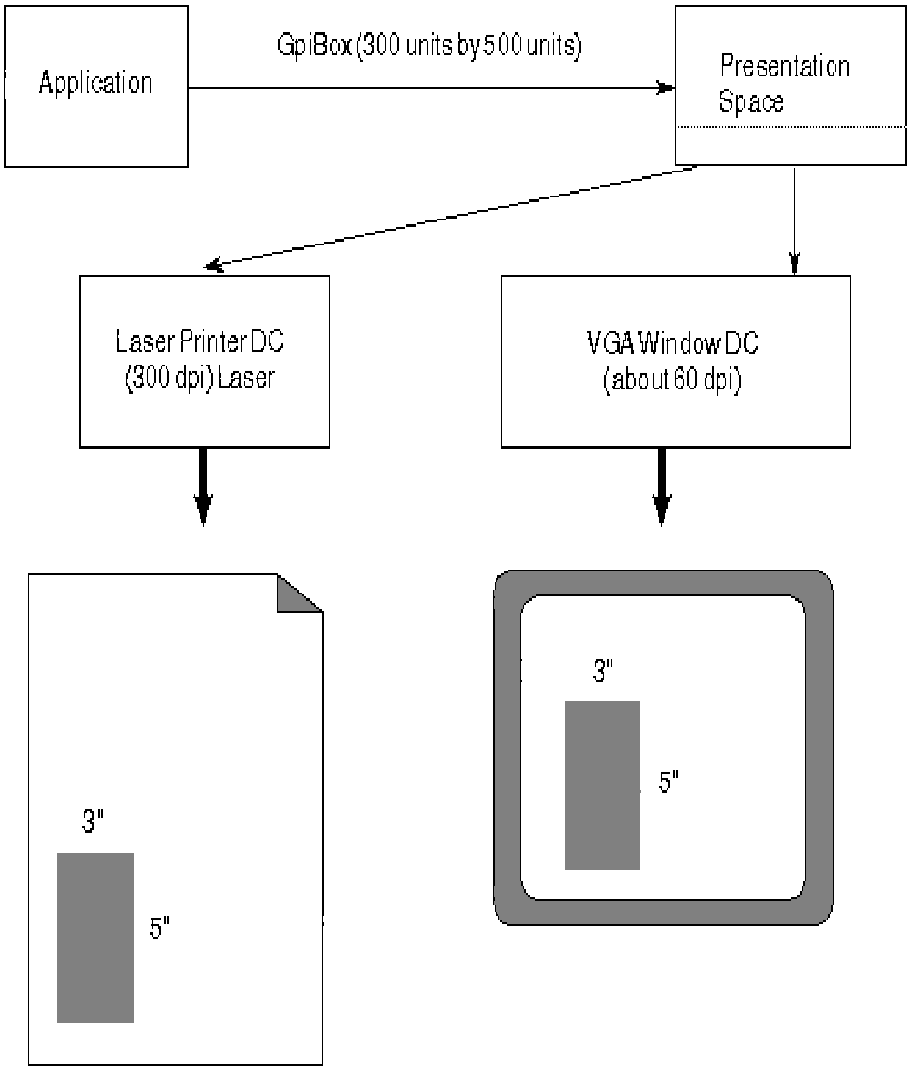
Device-Independence Considerations

The operating system supports true *WYSIWYG*. The same GPI functions the application used to create the picture or document on the display screen can be used to create the output on a printer. This is the major advantage of device independence.

An application must be designed around the options provided by the PM to ensure device independence, as a display (VGA) typically is 96 dpi, while a printer typically is 300 dpi.

Following are some application design considerations:

- Use DevQueryHardopyCaps to determine the horizontal and vertical width of the form and the maximum printable area.
- Create a presentation space with a presentation page size of 0 to ensure that the maximum window size and maximum page size are used.
- Use device-independent coordinates, such as LO_ENGLISH, or TWIPS. If you program in pels, transferring your output from screen to hard copy will compress the image, as shown in the following figure.



An Effect of Programming in Pels

Reassociating a presentation space defined in pels or other device dependent units can affect the dimension of the picture. If the pels are not square on the display device, the shape and scale (aspect ratio) will be different as well.

Starting a Print Job

The application signals the beginning of a print job by using DevEscape with the escape parameter DEVESC_STARTDOC. The application must specify a document name; usually this is related to the name of file being printed. Any GPI calls made before this DevEscape (DEVESC_STARTDOC) are ignored.

Querying the Device Capabilities

DevQueryCaps can be used to provide over 40 pieces of information about a specific device-for example, default character box size, horizontal and vertical resolution, or number of device-specific fonts.

Knowing that the device has only 1-bit color support (monochrome) might influence the application to use line style and patterns instead of colors for output.

Formatting the Page

Before print processing actually starts, it is recommended that an application's modal dialog be displayed, showing page progress and enabling the user to cancel the print job. If the user presses **Cancel** on this dialog, the application must call DevEscape with the parameter *DEVESC_ABORTDOC* . This allows the spooler and printer driver to clean up. A job is not created in the queue.

Usually the output on a page is generated by a series of GPI calls from the application. By calculating the size of character strings or graphics written to the presentation space, the application must be able to decide when to move to the next page in a multi-page document.

Starting the Next Page

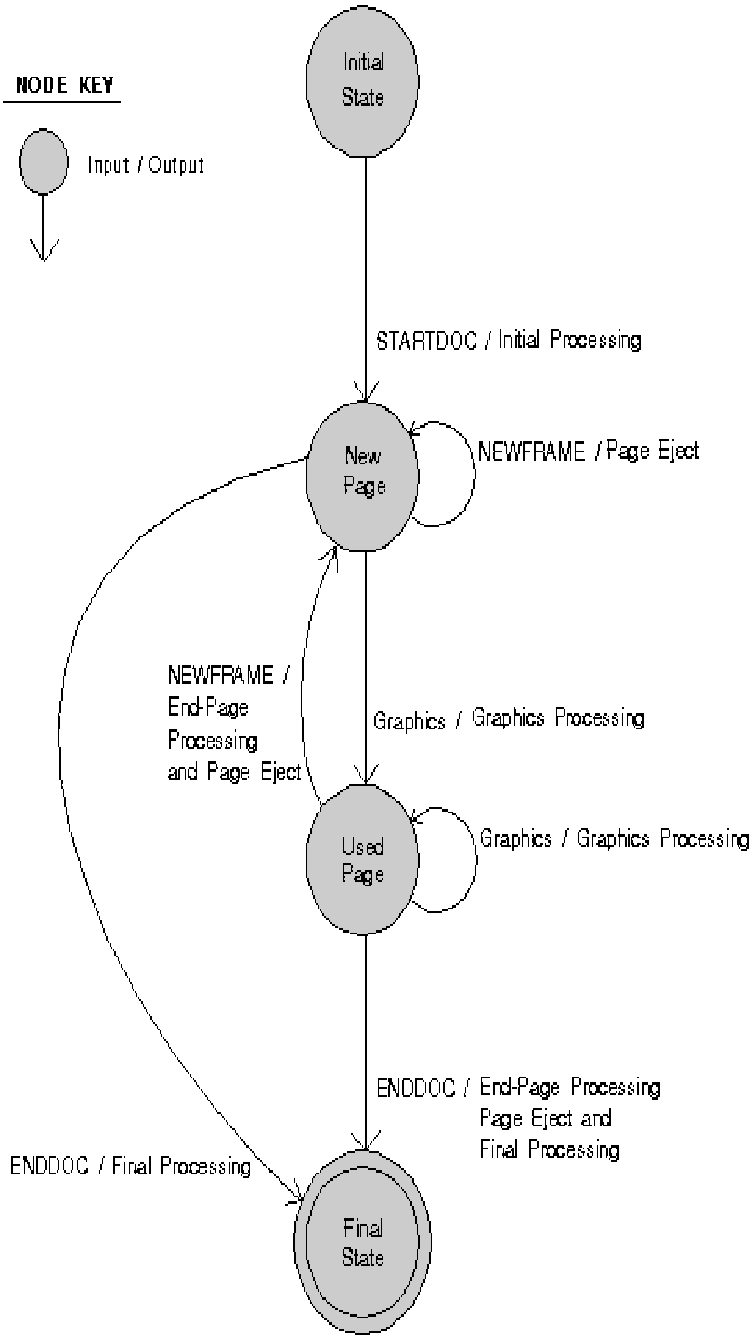
When the application has finished processing a page, it must call DevEscape with the *DEVESC_NEWFRAME* parameter. The progress dialog must be updated. The DevEscape *DEVESC_NEWFRAME* can be used to generate blank pages. The printer driver always issues a page eject whenever this DevEscape is received.

Issuing the *DEVESC_NEWFRAME* escape does not reset any attributes or fonts. However, the bounds and any clipping regions are reset.

Ending the Print Job

The end of a print document is signalled by the application's calling DevEscape with the *DEVESC_ENDDOC* parameter. The spooler returns a job identifier that can be used for job manipulation.

The following figure shows the processing carried out by a printer driver as it receives device escapes and graphics from the GPI.



Flow for Device Escapes

Note: Each device-escape pair of *DEVESC_STARTDOC* and *DEVESC_ENDDOC* creates one print job in a spooler queue. If the application needs to create multiple print jobs, use the following sequence:

```
DevOpenDC
GpiCreatePS
DevEsape(DEVESC_STARTDOC)
  .
  .
  .
DevEsape(DEVESC_ENDDOC)
DevEsape(DEVESC_STARTDOC)
  .
  .
  .
DevEsape(DEVESC_ENDDOC)
```

```
DevCloseDC
```

Disassociating the Presentation Space

When the print job is complete, the presentation space can be disassociated with the printer device context using GpiAssociate with a *NULL* parameter. Then the presentation space can be reassociated with the display device context.

Closing the Device Context

A device context is closed using DevCloseDC. The print thread must signal print termination to the main processing loop. The main loop can terminate the print thread and dismiss the Page Progress message box.

An application can display a message box to the user indicating a successful print and show the job ID returned by the spooler.

Print-to-File Considerations

Print-to-file means that the print data destined for a printer is stored in a file instead. The advantage of doing this is that the file can be used later, possibly on a different machine or different environment, and the data printed without requiring the original application.

The system provides three ways to print to file:

- The user can specify **Print to file** on the **Output** settings page of a printer object. When a print is requested, the system displays a dialog in which the user can enter a file name.

 This method is preferred because the application does not have to be aware; the user has full control of the system.

- The application can specify OD_DIRECT on a DevOpenDC call. The *pszLogAddress* field of the DEVOPENSTRUC structure specifies the file name as shown in the following example:

```
C:\TMP\MYPRINT.DAT
\\SERVER\DISK1\MYPRINT.DAT    -- UNC file name
\\PIPE\PRTPIPE1               -- A pipe
LPT1                          -- A port device
```

- The application can create the printer-specific format data and write it to a file using DosOpen.

Network Printing Considerations

The architecture of PM requires a printer driver during the print job creation process to deal with queries, such as DevQueryHardcopyCaps, and to provide the job properties dialog by way of DevPostDeviceModes. To do any printing across a network, a locally installed printer driver is required. In some cases, such as when the network server does not run PM, this is an advantage, in that all the conversion to printer-specific commands can be done on the requestor.

The printer object in the workplace provides seamless access to network printers; the user need only install the appropriate printer driver.

The application programmer also is helped. A network printer that is accessed by the user has a hidden shadow on the requestor. The job properties for the shadow printer object can be altered and are stored on the requestor. This enables the user to configure one or more variations on the original configuration without requiring intervention from a system administrator to create many network printer objects, each with a slightly different default job property configuration.

The shadow printer object is created on the requestor by creating a local queue and local device with no port connection. The application can enumerate these queues using SplEnumQueue, as before. The only difference is in the PRQINFO6 structure; there are two extra fields not in the PRQINFO3 structure. These two extra fields, named *pszRemoteComputerName* and *pszRemoteQueueName* contain the name of the remote server and the remote queue. The spooler uses these fields to automatically reroute print data, submitted by an application to the local shadow, to the appropriate network queue.

The printer object creates a local shadow only when the network printer object has been used or modified, because there could be a large number of network printer objects, and the spooler does not know which one the user wants to use. The local shadow of a network printer object is created in the following cases:

- When a file is dragged and dropped on the network printer object
- When the network printer object is moved, copied, or shadowed outside its original server folder
- When the printer or job properties are modified

Drag/Drop Protocol Considerations

When an application data file is dropped on a printer object, the application can receive a DM_PRINTOBJECT message if the application is running currently.

One DM_PRINTOBJECT parameter gives a DRAGITEM structure that describes the object that was dropped. The other parameter gives a PRINTDEST structure that contains all the parameters required to call DevPostDeviceModes and DevOpenDC.

The PD_JOB_PROPERTY flag indicates to the application that the user has requested a job properties dialog before printing. If this flag is not set, the application must not display the job properties dialog; it must use the job properties passed in the PRINTDEST structure.

The rest of the printing process follows the procedure described in Submitting a Queued Presentation Manager Print Job.

PMPRINT/PMPLOT Queue Processor Parameters

How the queue processor processes a print job is controlled by the optional queue-processor parameters that you can specify in the *pdopData* parameter of DevOpenDC. The PM adds these parameter values to the spool file.

The PMPRINT/PMPLOT queue processor parameters enable an application to do the following:

- Specify the number of copies of a print job
- Restrict printing to a specific area on the page
- Specify which part of a picture is to be printed
- Specify color output (if your printer allows this)
- Specify the foreground and background colors in a monochrome print

Note: The application need not use queue processor parameters if all that is required is a single copy of a picture, scaled to fit the page. These are the default settings of the queue-processor parameters.

The first parameter (*COP*) is used for all spool-file formats. The remaining parameters are valid for PM_Q_STD spool files only. Because PM_Q_STD data are used mainly for *graphic* data, these parameters are described in relation to the printing of picture files.

The PMPRINT/PMPLOT queue-processor parameters are separated by spaces and are:

COP=**n**

The *COP* parameter specifies the number of copies of the spool file that you want printed. The value of **n** must be an integer in the range of 1 through 999.

The default is *COP*=**1**.

ARE=C | **w,h,l,t**

The *ARE* parameter determines the size and position of the output area. This is the area of the physical page to which printing is restricted.

The default value of *ARE*=**C** means that the output area is the whole page. Note, however, that the printer cannot print outside its own device clip limits.

To size and position the output area at a specific point on the page, use *ARE*=**w,h,l,t**, where:

w, h	are the width and height of the desired output area
l, t	are the offsets of the upper-left corner of the output area from the left (l) and from the top (t) of the maximum output area

These four values must be given as percentages of the maximum output dimensions. The maximum output area is the area within the device clip limits.

FIT=S | **l,t**

The *FIT* parameter determines which part of the picture is to be printed. You can request the whole of the picture, scaled to fit the output area; or you can position the picture (actual size) anywhere within the output area. This could mean that the picture is clipped at the boundaries of the output area.

The default value of *FIT=S* causes the output to be scaled until the larger of the height or width just fits within the defined output area. The aspect ratio of the picture is maintained.

To print the picture in actual size, use *FIT*=**l,t**, where **l,t** are the coordinates of the point in the picture that you want positioned at the center of the output area: **l** is measured from the left edge of the picture, and **t** is measured from the top edge. The coordinates must be given as percentages of the actual dimensions of the picture.

XFM=0 | 1

The *XFM* parameter enables you to override the picture-positioning and clipping instructions that are provided by the *ARE* and *FIT* parameters, including their defaults.

The default value of *XFM*= allows the appearance of the output to be determined by the settings of the *ARE* and *FIT* parameters.

A value of *XFM*=**0** yields output as specified in the picture file. For example, applications that use many different forms can define different positions on each form for their output.

COL=M | C

The *COL* parameter enables you to specify color output if you have a color printer.

A value of *COL* =**M** creates monochrome output (black foreground with no background color). This is supported by all devices.

A value of *COL* =**C** creates color output. If you request color output on a monochrome device, the printer presentation driver tries to satisfy your request, which can cause problems because the only color available is black. For example, if the picture file specifies a red line on a blue background, both are drawn in black.

The default is *COL* =**M** when you are addressing a monochrome printer and *COL* =**C** when you are addressing a color printer.

MAP=N | A

The *MAP* parameter enables you to decide how the *neutral* colors (those that are not specified in the picture file) are printed.

The default value of *MAP*=**N** yields a *normal* representation of the screen picture on a printed page, which means that the page background is white and the foreground is black.

A value of *MAP*=**A** provides the reverse of the normal representation: the background is black and the foreground is

white on the printed page..

Note:

Pictures from a spool file can be printed differently than those printed when the spooler is not selected, because the default setting without the spooler is to print pictures their actual size, whereas the default for the queue-processor *FIT* parameter is "scaled to fit" the output area.

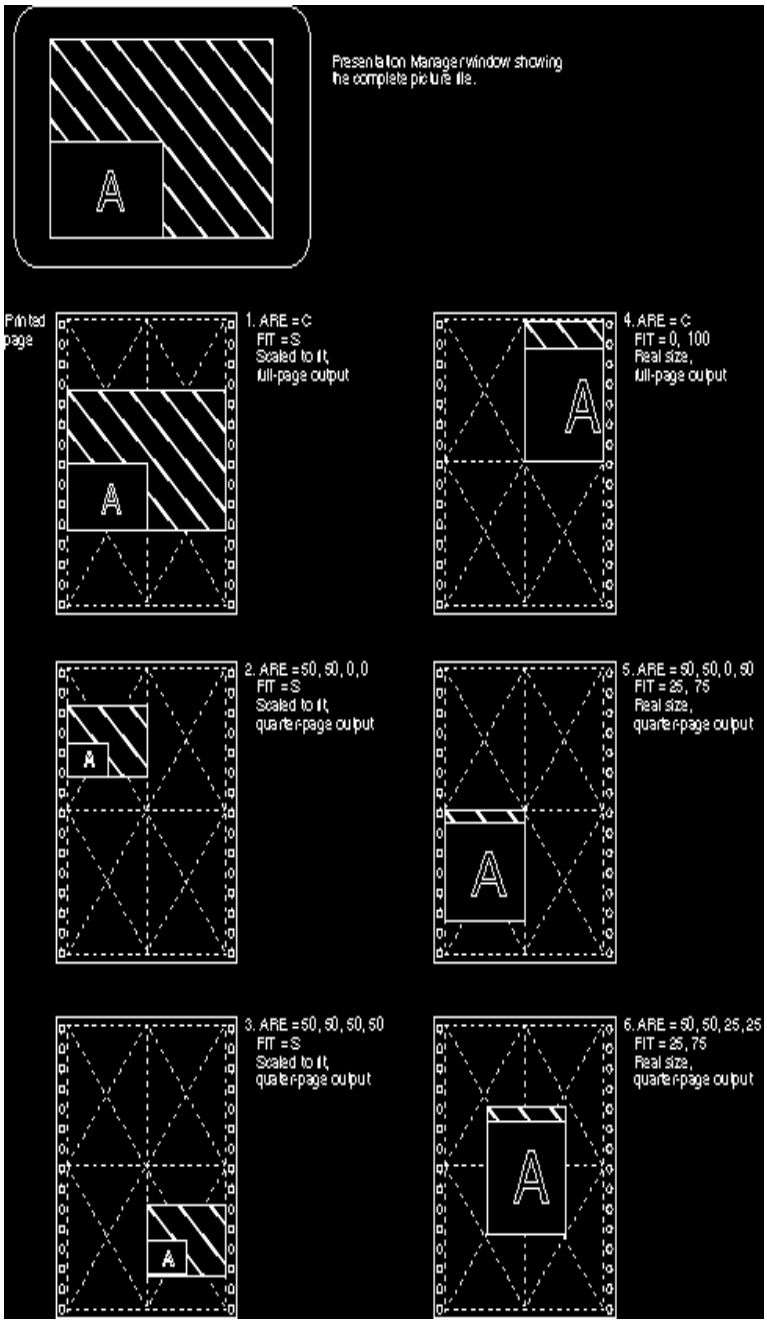
The queue-processor parameters must be separated by one or more blank spaces; for example,
"COP=3 ARE=C FIT=S".

The parameters can be listed in any order. Default values are used for parameters that are omitted or entered incorrectly.

Bit-map or image data (data constructed by setting pels on or off) is not affected by the *ARE* and *FIT* parameters. Consequently, if your picture contains bit-map or image data, the printed version will be different from the screen version if you use these parameters.

Examples Using the ARE and FIT Parameters

The application uses the *ARE* and *FIT* parameters to determine which part of the picture is printed, and where on the physical page it is printed. The following figure provides practical examples of how to use these parameters.



Examples Using the ARE and FIT Parameters

Submitting a Direct Presentation Manager Print Job

PM *direct printing* is done through a device context of type OD_DIRECT, as follows:

1. Output from the PM printer driver goes directly to the printer instead of to the spooler.
2. The application must specify **OD_DIRECT** in the DevOpenDC function.
3. The *pszLogAddress* field of the DEVOPENSTRUC structure is a port name (for example **LPT1**). The port name is held in the

pszLogAddr field of the PRDINFO3 structure returned by SplEnumDevice or SplQueryDevice.

4. All fields following the *pdriv* field in the DEVOPENSTRUC structure are ignored by the system.

The remaining steps in the printing process are the same as those for OD_QUEUED device contexts.

Submitting a Print Job Directly to the Spooler

The spooler provides functions that enable applications to submit print data directly to a spool queue. Normally, these functions are used by printer drivers to add a print job to a spool queue. However, this is not a recommended method unless the application is special-purpose.

The advantage of submitting print data directly to a spool queue is to bypass the GPI presentation layer. This can be useful, particularly, for sending print jobs to a network printer that is on a server that does not run the OS/2 operating system. However, there are certain requirements for direct spooling:

- Because the data bypasses the printer presentation driver, it must be in a format the printer can understand. Therefore, the application must be aware of and send the printer-specific commands.
- If the spooler is not active, the print jobs never will be printed.

The following steps are required to create a print job using the spooler directly:

1. Use SplQmOpen to open the PM. (Functionally, this is similar to using DevOpenDC, which is detailed in Opening a Queued Device Context.) You must specify a *queue name* of PMQOPENDATA for the logical address element, **pszLogAddress**. PMQOPENDATA is a data structure, identical to DEVOPENSTRUC, to be used with the SplQmxxx functions.
2. Use SplQmStartDoc to signal the start of your document. (This is same as using DevEscape with DEVESC_STARTDOC as used in PM printing.)
3. Use SplQmWrite to write data to the spool file.
4. Use SplQmEndDoc to signal the end of your document. (This is the same as DevEscape with DEVESC_ENDDOC as used in PM printing.)
5. Use SplQmClose to close the PM. (Functionally, this is similar to DevCloseDC.)

Spooler Management and Configuration

The spooler provides a set of functions that allows management of all aspects of the print subsystem. There are five categories:

- Job querying and manipulation (Spl*xxx* Job)
- Queue creation, querying, and manipulation (Spl*xxx* Queue)
- Device creation, querying, and manipulation (Spl*xxx* Device
- Listing ports, printer drivers, queue processors (SplEnum*yyy*)
- Listing printers across a network (SplEnumPrinter)

The symbol *xxx* typically represents Add, Delete, Enum, Query, and Set. For jobs and queues, *xxx* also includes Hold and Release. For devices, *xxx* includes Control. The symbol *yyy* is either Port, PrinterDriver, or QueueProcessor.

It is anticipated that most applications will not use any spooler functions at all except SplEnumQueue, SplQueryQueue, and SplQueryDevice. However, there are some specialized applications that might use the spooler functions for the following:

- An administrative tool to manage queues and jobs
- An automatic printer object configurator

 A new printer object can be created by using SplCreateDevice and SplCreateQueue. The print subsystem recognizes that this has occurred and puts a printer object directly on the desktop.

When the spooler is disabled, the following spooler functions do not return any results:

SplControlDevice
SplCopyJob
SplDeleteJob
SplEnumJob
SplQueryJob
SplReleaseJob
SplSetJob

No function is available to enable or disable the spooler under application control; this is a user decision. It is recommended that the spooler always be enabled.

Print Job Management

Some sophisticated applications might need to manipulate jobs in the following ways after they have been spooled into a queue:

- Modify the job parameters (for example, copies); use SplSetJob.
- Increase the priority of a particular job; use SplSetJob.
- Delete a job because a job based on newer data was created; use SplDeleteJob.

Presentation Spaces and Device Contexts

A presentation space is a data structure maintained by the operating system in which information relevant to the graphic output is stored. The information is related to both the subsequent drawings (such as colors or line styles) and the presentation space resources (such as color tables or fonts).

The first task in a graphics application is to define a presentation space, because so many of the GPI functions must operate within them. A presentation space is required for each output device currently in use by your application, including each separate window on the display screen. In almost all cases, the presentation space is set up to be *device-independent*, because the requirements of each possible output device are so different.

Note: In some cases, it is possible to disassociate a presentation space from one device and associate it with another, thereby allowing the presentation space to be shared.

To facilitate the device independence of the PM programming interface, all device-specific information is held in a device context. A device context is a data structure that identifies a particular instance of an output device and contains all the device-specific information, such as the logical name of the device and the presentation driver name. Each separate instance of an output device that you intend to use must be described in a device context. For example, if a single application uses more than one window, each of those windows must have its own window device context.

The applicable device context, then, must be *associated* with the presentation space in order to send graphics data to that output device.

This chapter describes presentation spaces and device contexts. The following topics are related to the information in this chapter:

- Graphics primitives
- Coordinate spaces and transformations
- Graphics segments

About Presentation Spaces

There are three types of presentation spaces:

- Standard micro
- Cached micro
- Normal

Micro Presentation Spaces

You need a *micro presentation space* if your application creates a separate presentation space for each output device *instance* and if its output is a simple drawing.

Although run time memory requirements vary according to the graphics function an application uses, a typical micro presentation space graphical application uses 315KB of GPI, engine, and display memory resource (code and data). A micro presentation space cannot be reassociated with a different device.

Standard Micro Presentation Space

Use a *standard micro presentation space* for drawing on any type of output device, provided the presentation space is not in *retain mode* or *retain and draw mode* .

There are two different functions you can use to access a standard micro presentation space. The following table lists both functions and their considerations.

Function Name	Usage	Closing Function
GpiCreatePS	Accepts an anchor handle, a device-context handle, and a presentation space size as parameters. Creates both normal presentation spaces and micro presentation spaces.	GpiDestroyPS destroys a presentation space and releases all resources owned by the presentation space.
WinGetScreenPS	The presentation space represents the entire display screen. Warning: Exercise caution when using this function as the graphic output can overlap individual windows.	WinReleasePS

Cached Micro Presentation Space

The window manager maintains a cache of micro presentation spaces for windows on a display screen. The cache is provided for applications that use a large number of windows, and where each window requires a temporary presentation-space device-context pair for a short sequence of output operations. These presentation spaces belong to the system rather than to your application and are allocated only on a temporary basis.

Cached micro presentation spaces are provided by the window system rather than by the GPI. Their use is synchronized with other window activities. For example, you need not associate a cached micro presentation space with the display screen. The window manager does this

for you.

Cached micro presentation spaces offer the best system performance because, unlike normal presentation spaces and standard micro presentation spaces, they are not permanently allocated to an application. However, cached micro presentation spaces can be cumbersome to use because all the attributes must be initialized continually.

Use a cached micro presentation space to send output only to a window on the display device. There are three different functions you can use to access a cached micro presentation space, each with its own considerations. These functions are listed in the following table.

Function Name	Usage	Closing Function
WinBeginPaint	Accepts a NULL presentation space handle for a cached micro presentation space. The presentation space created by this function is already preassociated with the window device context, making this the easiest function to use. Usually this type of creation is in response to a WM_PAINT message.	WinEndPaint automatically releases the presentation space, no matter what type.
WinGetScreenPS	The presentation space represents the entire display screen. Warning: Exercise caution when using this function as the graphic output can overlap individual windows.	WinReleasePS
WinGetPS	The presentation space can represent the entire desktop, or any other window. The presentation space can be used to process any message, but it must be returned to the cache when message processing is complete.	WinReleasePS

In general, use a cached micro presentation space to process a single paint message when no presentation space information needs to be remembered between messages. The presentation space must be both obtained and released during the processing of that message. All application information stored in a cached micro presentation space is lost as soon as it is released to the cache.

You must provide a window handle on input to WinBeginPaint and WinGetPS. The resulting presentation space is defined specifically for that window, and cannot be reassociated.

The cached micro presentation space is always:

- Defined in *pels* -although you can change the units using GpiSetPS
- Formatted GPIF_LONG
- Given a suitable size by the system.

When you finish using a cached micro presentation space, you do not have to disassociate it from the window device context because WinReleasePS or WinEndPaint performs the disassociation. This makes the cached micro presentation space available for use in other windows. The presentation space itself cannot be deleted.

Cached micro presentation spaces are used serially. The next time you need a cached presentation space, access a new one using the appropriate function. Each time you get a cached micro presentation space, graphics attributes are reset to their default values.

Normal Presentation Spaces

You must use a *normal presentation space* if you require your application to use the same presentation space to send output to multiple devices (a display screen and printer, for example), or if it uses the segment and retained-drawing functions to generate complex drawings.

There is only one function you can use to create a normal presentation space:

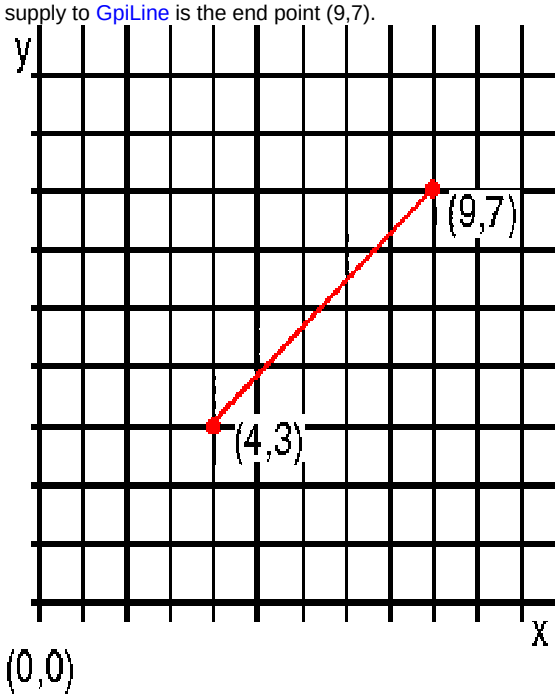
Function Name	Usage	Closing Function
GpiCreatePS	Accepts an anchor handle, a device-context handle, and a presentation space size as parameters. Creates both normal presentation spaces and micro presentation spaces.	GpiDestroyPS destroys a presentation space and releases

If a normal presentation space is used, or if *metafiling* is carried out, more run time memory is used than if a micro presentation space was used. A normal presentation space requires 114KB more than a micro presentation space.

Modal Graphic Systems

The graphic output sent to the presentation space is created by graphic primitives, such as straight or curved lines. How those primitives appear depends in part on the *mode* of the presentation space.

In modal systems, certain information that modifies a graphic primitive is established before you use the instruction to draw the primitive. The following figure is an example of a red, solid line being drawn from (4,3) to (9,7). With PM's modal interface, the only value you need to supply to GpiLine is the end point (9,7).



Line Drawn with GpiLine

The line is drawn automatically with the current color, red, in the current line type, solid, and starting from the current position, (4,3). There are separate functions for specifying the current color, line type, and position.

In nonmodal graphics systems, you supply information relevant to an instruction when you use that instruction. For example, you could use a line-drawing instruction in which you would specify that the line is to be green, dotted, and drawn from point *a* to point *b*. Those values or *attributes* are bound to a particular drawing instruction, and affect no other. If you are going to continue the line from point *b* to point *c*, you must again specify "green", "dotted", "point b", and finally, "point c".

For the PM programming interface, the default values for the graphics primitives (or current values if the defaults have been modified) are stored in the presentation space along with the current position. While at first it might seem like many functions are necessary to perform one step, such as drawing a line, a modal graphics system actually saves resources. After you perform the first line draw, chances are that if you draw a second or third line, the line attributes are not likely to change; that is, the mode of the presentation space remains the same.

Thus, the constant repetition of attributes in a nonmodal graphics system actually uses more resources if the desired output requires more than one or two graphics functions.

It is possible to define graphics, and store their definitions, without sending them to a screen or a printer. In these situations, the concept of *attribute currentness* becomes relevant. The color of a line on a screen, for example, is the color that was current when you defined the line. This is not always the color that is current when the line is drawn.

Current Position

The *current position* is the *world-coordinate space* position at which the next drawing request begins. It is part of the data stored in the presentation space. When you create or obtain a presentation space, the current position is set to the origin (0,0). The current position also is reset to (0,0) whenever a presentation space is associated with a different device context.

Most drawing requests update the current position. If you draw a line from the origin to (3,5), for example, the current position is updated to (3,5). The current position also can be updated explicitly by GpiSetCurrentPosition or GpiMove. If you do not want the first drawing in a presentation space to start at (0,0), call GpiSetCurrentPosition first.

Primitive Attributes

The graphics primitives and the functions that draw them are discussed in Line and Arc Primitives through Fonts. The primitive's function allows the attributes of the primitive to be set for all subsequent use of that primitive. The attributes are stored in data structures called *bundles*. There are functions that allow you to set the attributes of the primitives before the drawing of any primitive is performed. GpiSetAttrs allows you to not only specify the primitive for which the attributes are going to be set, but also, the settings for the specific attributes. GpiQueryAttrs returns the current values of specified attributes for specified primitives.

Two other functions, GpiSetAttrMode and GpiQueryAttrMode, enable you to define or query the attribute mode settings, AM_PRESERVE and AM_NOPRESERVE. The AM_PRESERVE setting preserves the value of an attribute any time that attribute is changed. The previous value can be recovered easily if, for example, you need six dotted lines, one solid line, then six more dotted lines. The AM_NOPRESERVE setting discards the previous value of an attribute once the value is changed. The values of the attributes are presentation space resources.

Using Presentation Spaces

You can use presentation-space functions to:

- Create a normal, micro, or cached micro presentation space
- Delete, save, or restore a presentation space
- Define or determine the presentation space attributes

Obtaining and Creating a Presentation Space

Normal presentation spaces and micro presentation spaces are created using GpiCreatePS. Cached micro presentation spaces are created using WinBeginPaint or WinGetPS.

Deleting a Presentation Space

A presentation space consumes a considerable amount of memory, so you should delete it when your application no longer requires it. Use

GpiDestroyPS to delete either a normal or micro presentation space.

When you finish using a cached micro presentation space that you accessed with WinGetPS, release it with WinReleasePS. You need not delete a presentation space that you accessed with WinBeginPaint. PM does this for you when you call WinEndPaint.

Saving and Restoring a Presentation Space

You can save certain attributes and resources of a presentation space, modify its fields, draw in the modified presentation space, then restore it with GpiSavePS and GpiRestorePS. When you call GpiSavePS, the graphics engine copies the following items from the current presentation space onto a special stack:

- Primitive attributes
- Transformation matrixes
- Viewing limit
- Clip path
- Clip region
- Current position
- Loaded logical color table
- Loaded logical font

You can push the contents of a presentation space on the stack as many times as is necessary. GpiRestorePS pops the contents of a presentation space off the stack.

Presentation Spaces Review

The following table summarizes the features and restrictions of each type of presentation space.

Presentation Space Features and Restrictions

Feature/Restriction	Normal	Standard Micro	Cached Micro
Device types supported	Any device	Any device	Video-display window only
Number of supported devices	Multiple	One	One
Association	Associate and disassociate as required.	Associate at creation; cannot disassociate.	N/A.
Retained graphics	Yes	No	No
Available GPI functions	All functions	All except segment functions	All except segment functions
Memory considerations	Highest memory usage	Medium memory usage	Quickest allocation

About Device Contexts

Device contexts link presentation spaces to devices by converting device-independent presentation space information into

device-dependent information. This conversion occurs in the *presentation driver* , a low-level program that is transparent to the application. A presentation driver is device-driver code that converts the application commands into output commands specific for each output device.

After creating a device context, it must be associated with a presentation space before the drawing (or primitives) can be printed or displayed. When a presentation space and a device context are associated with each other, any output you direct to the presentation space is directed automatically at the device context and displayed on the physical device. For example, before you can draw a picture in a display window, the picture's graphics presentation space must be associated with the window's device context.

Device contexts also give applications access to important device information such as screen dimensions or printer capabilities.

As with presentation spaces, there are three types of device contexts:

- Cached
- Window
- Normal

Cached Device Contexts

A *cached device context* is one that already is associated with a cached presentation space when that space is obtained from the cache.

Window Device Contexts

Window device contexts are a special type, in that they are the only device contexts obtained with WinOpenWindowDC. As their name implies, the output device they represent is a display window. WinOpenWindowDC accepts a window handle as input and returns the window device context handle. The window device context handle can then be associated with a standard micro presentation space or a normal presentation space.

The window device context is closed automatically when its associated window is deleted.

Normal Device Contexts

A *normal device context* (also called a *standard* or *noncached* device context) links a presentation space with any nonwindow output device. Use DevOpenDC to obtain a device context handle to one of the six normal device contexts listed in the following table. Each of the six normal device contexts must be closed by DevCloseDC before your application is closed.

DC Type	Purpose	Usage
Queued	Links a presentation space with a printer or plotter shared by multiple applications sending spooled print jobs to the print queue. Queued device contexts store print jobs by using a program called *print spooler*, which keeps track of the order in which the jobs arrive at the printer and in which they are printed.	Applications use queued device contexts to offload printing control from the application.

Direct	Links a presentation space with a printer or plotter, directly bypassing the spooler and print queue. A direct device context is used by the spooler to process jobs as they are removed from the print queue.	Applications normally do not use direct device contexts, unless they are avoiding the queue (for security reasons) or going directly to a dedicated machine.
Information	Links a presentation space with a printer or plotter, directly enabling device information to be queried, but producing no output on the device.	An application can use an information device context with lower memory overhead, rather than use a direct device context, which could provide the same information.
Memory	Links a presentation space with a bit map.	Applications use memory device contexts for drawing to the bit map and using it as a source or target of BitBlt operations.
Metafile	A special device context that enables a picture output to its associated presentation space to be recorded in a metafile for interchange for future use.	Only applications that use metafiles use metafile device contexts.
Metafile_NoQuery	Functionally identical to metafile; however, querying of presentation space attributes is not allowed.	If attributes of the presentation space are not to be queried, this device context offers improved performance over metafile.

Using Device Contexts

In addition to associating or disassociating presentation spaces with device contexts, you can use device context functions to:

- Obtain or create a device context
- Associate a device context with a presentation space
- Close a device context
- Retrieve information about a device's capabilities

Creating a Device Context

You can create a normal device context by calling DevOpenDC. This function requires that you specify one of the six normal types. It also requires that you pass certain device-initialization data, including a logical address, the device-driver name, device-driver data, a description of the device type, and information about the queue (if the device is a queued device). The device-initialization data is passed in a DEVOPENSTRUC structure.

The following figure is an example of this structure.

```
typedef struct _DEVOPENSTRUC {   /* dop                          */
    PSZ      pszLogAddress       /* Logical-device address       */
    PSZ      pszDriverName       /* Device-driver name           */
    PDRIVDATA  pdriv             /* Pointer to extra driver data */
    PSZ      pszDataType         /* Type of queued data          */
    PSZ      pszComment          /* Optional spooler info        */
    PSZ      pszQueueProcName    /* Queue-processor name         */
    PSZ      pszQueueProcParams  /* Queue-processor arguments    */
    PSZ      pszSpoolerParams    /* Spooler arguments            */
    PSZ      pszNetworkParams    /* Network arguments            */
} DEVOPENSTRUC;
```

The last four fields in this structure apply only to queued devices.

The following figure shows how to create a nondisplay device context for a printer.

```
HDC hdcPrinter;                    /* Handle of printer device context    */
HAB hab;                           /* Anchor-block handle                 */
DEVOPENSTRUC dop;                  /* Device information                  */

dop.pszLogAddress = "lpt1";        /* Logical-device address              */
dop.pszDriverName = "EPSON";       /* Device-driver name                  */
dop.pdriv = (PDRIVDATA) NULL;      /* Pointer to driver data              */
dop.pszDataType = "PM_Q_STD";      /* Standard queued data                */

hdcPrinter = DevOpenDC(hab,
                       OD_DIRECT,  /* Direct device type                  */
                       "*",        /* No data in OS2.INI                  */
                       4,          /* Use first 4 fields in dop structure */
                       (PDEVOPENDATA) &dop,
                       (HDC) NULL);
```

The following figure is an example of how to create a standard device context for a metafile.

```
HDC hdcMeta;                                /* Handles of metafile       */
                                            /*    and window DCs         */
HAB hab;                                    /* Anchor-block handle       */
DEVOPENSTRUC dop;                           /* Device information        */

dop.pszLogAddress = NULL;                   /* Logical-device address    */
dop.pszDriverName = "DISPLAY";              /* Device-driver name        */

hdcMeta = DevOpenDC(hab,
                    OD_METAFILE,            /* Metafile DC               */
                    "*",                    /* No data in OS2.INI        */
                    2,                      /* Use first 2 fields in dop */
                    (PDEVOPENDATA) &dop,    /* Structure for system info */
                    NULL)                   /* Compatible with screen    */
```

Associating Presentation Spaces with Device Contexts

Drawing graphic objects requires a presentation space and a device context to direct output to a specific instance of an output device, such as a display window or a printer. This *association* enables the device context to identify the output device for that presentation space. Further, the device context identifies the particular *instance* of the output device, such as a printer or display window.

A presentation space can be associated with only one device context at a time. The reverse is also true: a device context can be associated with only one presentation space at a time.

The following figure shows how a presentation space is associated with a window device context. It is then disassociated from the window device context and associated with a printer device context. It cannot be associated with both device contexts simultaneously.

```
WM_Create:
     hdcScreen = WinOpenWindowsDC (hwnd);
     phs = GpiCreatePS (...GPIA.Assoc);
      .
      .
      .

WM_COMMAND:

  Case IDM_File PRINT:                      /* Device selection        */
     hdcPrinter = DevOpenDC (...);
     GpiAssociate (hps, NULL);              /* Disconnect from screen  */
     GpiAssociate (hps, hdcPrinter);        /* Connect to printer      */
      .
      .                                     /* Output                  */
      .
     GpiAssociate (hps, NULL);              /* Disconnect from print   */
     GpiAssociate (hps, hdcScreen);         /* Reconnect to screen     */
      .
      .
      .

WM_PAINT:
     WinBeginPaint (hwnd, hps, NULL);
      .
      .                                     /* Output                  */
      .
```

The following figure shows how to open a window device context and associate it with a normal presentation space.

```
HDC hdcWin;            /* Window device-context handle     */
HPS hpsWin;            /* Normal-presentation-space handle */
HWND hwndClient;       /* Client-window handle             */
HAB hab;               /* Anchor-block handle              */
SIZEL sizlPage;        /* Presentation page                */

hdcWin = WinOpenWindowDC(hwndClient);
hpsWin = GpiCreatePS(hab, hdcWin, &sizlPage,
                     PU_LOENGLISH | GPIA_ASSOC);
```

Note: This type of code is used when the device context is defined before the presentation space.

WinOpenWindowDC can be called only once for a particular window and returns an error if called a second time. WinQueryWindowDC can be used to obtain a window device context previously allocated using WinOpenWindowDC. The following figure shows how to create a presentation space with page units of 0.01 inch (PU_LOENGLISH) and associate it with a printer device context. As input to GpiCreatePS, you supply the height and width of the presentation page.

```
HAB hab;          /* Anchor-block handle         */
HPS hpsPrinter;   /* Presentation-space handle   */
HDC hdcPrinter;   /* Device-context handle       */
SIZEL sizlPage;   /* Page structure              */
.
.
.
hpsPrinter = GpiCreatePS(hab, hdcPrinter, &sizlPage,
   PU_LOENGLISH | GPIA_ASSOC);
```

Closing a Device Context

To close a device context that your application opened with DevOpenDC, call DevCloseDC. However, you should not try to close a device context that you opened with WinOpenWindowDC. The operating system does this automatically when the associated window is deleted.

Determining Device Capabilities

Once you have created a device context for a particular output device, you can determine the capabilities of that device by calling DevQueryCaps. This function retrieves the following information:

- Device technology (whether the device is a *raster* or *vector* device)
- Maximized window dimensions (if the device is a video display)
- Page dimensions (if the device is a printer or plotter)
- Character-box dimensions
- Marker-box dimensions
- Pel resolution
- Color capabilities
- Mix-mode capabilities

You can use this information, for example, to select fonts, set up the presentation page, or create a new logical color table.

Paths

A *path* is a graphics object composed of one or more figures drawn with graphic primitives. A path can be used for defining a filled area, a complex clipping shape, an outline, and for drawing geometric wide lines.

The following topics are related to information in this chapter:

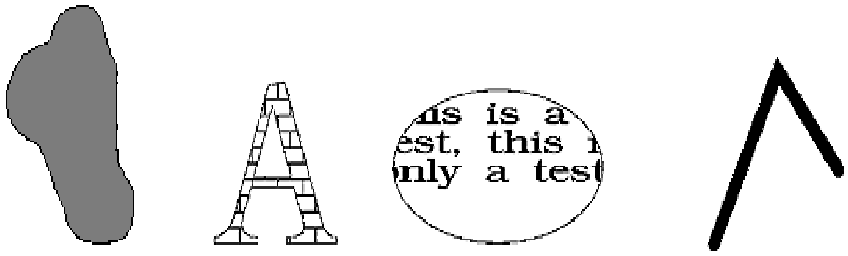
- Presentation spaces and device contexts
- Line and arc primitives
- Area primitives
- Color and mix attributes
- Regions
- Clipping

About Paths

Paths provide several useful drawing techniques including:

- An alternate means of generating filled or outlined irregular figures and nonrectangular shapes
- An efficient means of producing hollow text in an outline font
- The only means of *clipping* (restricting the area of interest of a picture) to circular, elliptical, or other nonrectangular figures
- The only means of drawing lines with a thickness that can be transformed

These techniques are respectively illustrated in the following figures:



Drawing Techniques Achieved with Paths

A path is similar to an area, but there are fundamental differences between the two. The following table describes their similarities and contrasts.

Paths	Areas
Functions that define a path are bracketed by GpiBeginPath and GpiEndPath.	Functions that define an area are bracketed by GpiBeginArea and GpiEndArea.
Paths are defined in world coordinates.	Areas are defined in world coordinates.
Paths can be modified before being displayed or used for clipping.	Areas are displayed immediately on completion.
Paths can be used by applications to perform clipping.	Areas cannot be used as clipping regions.
Paths can be converted into graphics objects called regions.	Areas cannot be converted into regions.
Six operations can be performed on paths, each requiring a separate function: Outline Fill Modify Stroke Convert to clip path Convert to region	Only two operations can be performed on areas (specified with options when creating the area): Outline Fill

Path Attributes

Paths do not have a separate ...BUNDLE data structure associated with them. Their attributes are defined by LINEBUNDLE and AREABUNDLE. There are certain line attributes that apply only to *geometric* lines, or lines created with some path operations. Depending on the purpose of a path, only certain path attributes are in effect. These and the other path attributes are described in this chapter.

When an application creates a presentation space, the path attributes are set to the default values shown in the following table.

Path Attribute Default Values

Attribute	Default Value	Function that Redefines Attribute

Geometric width	None	GpiSetLineWidthGeom
End	LINEEND_FLAT	GpiSetLineEnd
Join	LINEJOIN_BEVEL	GpiSetLineJoin
Width	1.0	GpiSetLineWidth
Type	LINETYPE_SOLID	GpiSetLineType
Color	CLR_NEUTRAL	GpiSetAttrs (LBB_COLOR)
Mix mode	FM_OVERPAINT	GpiSetAttrs (LBB_MIX_MODE)
Pattern symbol	Solid	GpiSetPattern
Pattern set	LCID_DEFAULT	GpiSetPatternSet
Reference point	(0,0)	GpiSetPatternRefPoint
Foreground color	CLR_NEUTRAL (black on most devices)	GpiSetAttrs (ABB_COLOR)
Background color	CLR_BACKGROUND (white on most devices)	GpiSetAttrs (ABB_BACK_COLOR)
Foreground mix	FM_OVERPAINT	GpiSetAttrs (ABB_MIX_MODE)
Background mix	BM_LEAVEALONE	GpiSetAttrs (ABB_BACK_MIX_MODE) Geometric width, End, and Join take effect only when a path is converted to a geometric wide line using GpiStrokePath or GpiModifyPath. Width, Type, Color, and Mix mode take effect only when a path is outlined using GpiOutlinePath. Pattern symbol, Pattern set, and Reference point take effect only when a path is filled using GpiFillPath or GpiStrokePath.

Line Width and Geometric Width for Paths

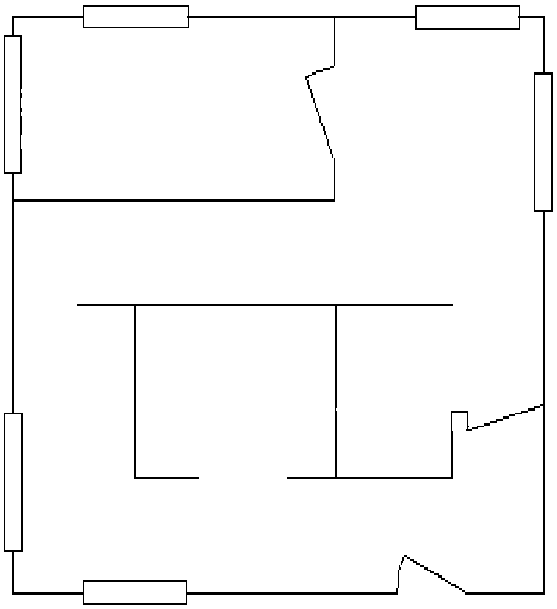
Cosmetic lines represent the mathematical ideal of a one-dimensional line. The width attribute associated with them is provided only for ease of drawing these lines larger than one pel.

In contrast, *geometric* line width is not an attribute; it is a geometric property. To set the width of a geometric line, an application can use GpiSetLineWidthGeom. This geometric width takes effect only when a path is converted to a geometric wide line using GpiStrokePath or GpiModifyPath.

If a geometric line is drawn before the geometric width is specified, the drawn line is defined by the cosmetic line width-usually 1-which results in the thinnest possible line for the currently associated device. A geometric line width has no default in the same way that the sides of a box, drawn with GpiBox, have no length until specified by the function.

The value specified with GpiSetLineWidthGeom is the thickness of the line in world coordinates, and it is subject to scaling by the current transformations in effect at the time GpiModifyPath or GpiStrokePath is called. For example, if you apply a transform with a scaling factor of 0.5, for an object whose current geometric line width is four world coordinates, the width of the displayed line will be halved.

In the floor plan shown in the following figure, the outer walls were drawn using geometric lines. All of the other objects were drawn using normal lines.



Geometric Lines and Normal Lines

Your application can determine the current geometric width by calling GpiQueryLineWidthGeom.

Line End

The line end attribute specifies the shape of the unattached end of a geometric line. Lines whose shapes are partially defined by a geometric width have to be closed, unlike cosmetic lines that simply end. An application can draw geometric lines with square, flat, or round ends.

Note: The end attribute takes effect only when a path is converted to a geometric width line using GpiStrokePath or GpiModifyPath.

Set the current geometric line end attribute with GpiSetLineEnd. The attribute applies to all subsequent unattached lines within the path bracket. The following table describes the three standard line ends provided by the PM programming interface; you cannot define your own end types.

Standard Geometric Line End Types

Type	Identifier	Long Value
Flat, flush with line end	LINEEND_FLAT	1L
Round, past line end	LINEEND_ROUND	2L
Flat, but extends past line end	LINEEND_SQUARE	3L

The default line end type (LINEEND_DEFAULT) is identical to the LINEEND_FLAT type, and has a long value of 0L. The error linetype (LINEEND_ERROR) has a long value of -1L.

The following figure illustrates these three types of line ends. Your application can determine the current geometric line end by calling GpiQueryLineEnd.



Closing Unattached Geometric Lines

A square line end extends the line by a distance that is half the width of the line. For example, if the line is six coordinate units wide, a square line end extends it by three coordinate units.

A round line end is constructed by drawing a circle whose radius is half the width of the line.

Line Join

The line join attribute specifies the shape formed by two intersecting geometric lines. Where one wide line joins another, the nature of the join must be defined. An application can select a beveled, rounded, or mitred line-join style.

Note: The join attribute takes effect only when a path is converted to a geometric wide line using GpiStrokePath or GpiModifyPath.

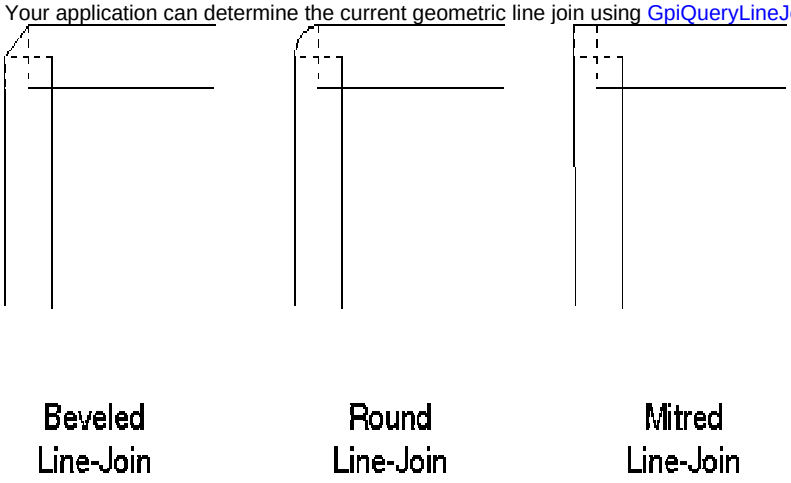
Set the current geometric line-join attribute with GpiSetLineJoin. The attribute applies to all subsequent intersection of lines within the path bracket. The following table describes the three standard line joins provided by the PM programming interface. You cannot define your own join types.

Standard Geometric Line Join Types

Type	Identifier	Long Value
Diagonal corner	LINEJOIN_BEVEL	1L
Rounded corner	LINEJOIN_ROUND	2L
90° angled corner	LINEJOIN_MITRE	3L

The default join type (LINEJOIN_DEFAULT) is identical to the LINEJOIN_BEVEL type, and has a long value of 0L. The error linetype (LINEJOIN_ERROR) has a long value of -1L. The following figure illustrates these three types of line joins.

Your application can determine the current geometric line join using GpiQueryLineJoin.



Joining Wide Lines

Cosmetic Line Attributes for Paths

When a path is drawn as a cosmetic line with GpiOutlinePath, the following attributes are defined by the LINEBUNDLE data structure:

- Line width
- Line type
- Line color
- Line mix

These attributes follow the current definitions and appear just as line primitives do. Since the line primitive has only a foreground color and mix attribute, the current color of the drawing surface affects the appearance of these paths more than it does the appearance of paths that define geometric lines. Those paths follow the area attributes and are affected by background color and background mix attributes as well.

Area Attributes for Paths

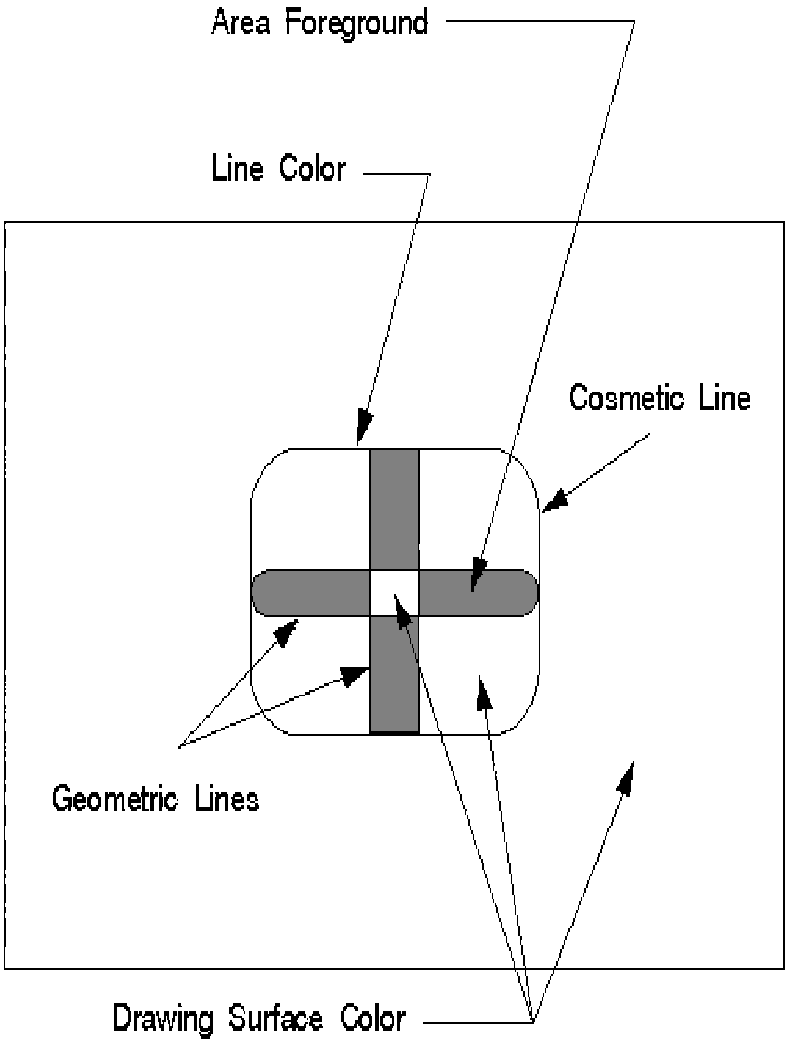
When a path is drawn as a geometric line with GpiStrokePath or GpiModifyPath, the following attributes are defined by the AREABUNDLE data structure:

- Pattern symbol
- Pattern set
- Pattern reference point
- Area foreground color
- Area background color
- Area foreground mix
- Area background mix

Area attributes follow the current definitions and appear just as area primitives do. The operating system uses the pattern symbol to fill the interior of the path that defines the geometric line. Any alterations to the fill pattern or reference point affect the appearance of a geometric line just as it does an area.

Path Color and Mix Attributes

The color attribute defines the color used to draw a primitive or an object. The mix attribute determines how the color of a primitive or an object is combined with the color of the drawing surface, or any other objects on the surface. Unlike the color and mix attributes for primitives, and certain other objects, the path color is defined by different attributes in different situations, as illustrated in the following figure.



Path Objects

Path colors are determined by the area color, area background color, area mix, and area background mix attributes if the path is stroked into a wide line. The cosmetic line path color is determined by the line color and mix attributes.

The line color defines the color used to draw the output from any of the IBM OS/2 line functions called within the path. The line colors can change within the path bracket, just as they can within area brackets. The interaction of the line with the drawing-surface color is controlled by the line mix attribute, which also can vary within the path bracket.

After the path is closed, the path can be *stroked* into a geometric (or wide) line. Depending on the construction options, the geometric line can be filled. The appearance of the pattern symbol used to fill the path depends on the area attributes. The definitions of the area color and background color depend on the pattern symbol.

Path Brackets

The functions that create and define the path always are bracketed by the GpiBeginPath and GpiEndPath functions. Only one path can be defined between these two functions. However, within the path, there can be any number of figures.

GpiBeginPath signals the start of a path definition. Each path you define becomes the *current path* and replaces any existing path in the presentation space. The *current position* is not changed by GpiBeginPath, although it is updated by any drawing instructions that follow.

GpiBeginPath accepts as input only a *path identifier* of 1. This identifier is used by the functions performing path operations to identify the path.

Due to the number of different operations that can be performed on paths, the path definition can contain both open and closed figures. If a figure does not start and end at the same coordinate point, it is classified as an open figure. Unlike an area bracket, if you use GpiSetCurrentPosition or GpiMove within a path definition, the current figure is not automatically closed.

The PM provides a method of closing figures within a path. GpiCloseFigure (valid only in path brackets) closes a figure in the path whose start and end points are not the same, by drawing a straight line from the current position to the start position of the figure. Use GpiCloseFigure if you want the line join attribute to be applied between the start and end point of a figure. Do not use it if you want the line end attributes to be used as the start and end points for a geometric wide line.

If the path definition contains full arc or box primitives, however, you must not close them using GpiCloseFigure. GpiBox and GpiFullArc generate completely closed figures and must not be used within another figure definition.

To signal the end of the current path definition, use GpiEndPath. The path definition is constructed in the currently associated presentation space. The current position always is updated to the end point of the last line drawn.

The functions that generate a path are enclosed in a path bracket. Applications can use the functions in the following list within a path bracket.

- GpiBeginElement
- GpiBox
- GpiCallSegmentMatrix
- GpiCharString
- GpiCharStringAt
- GpiCharStringPos
- GpiCharStringPosAt
- GpiCloseFigure
- GpiComment
- GpiCreateLogFont
- GpiDeleteSetId
- GpiElement
- GpiEndElement
- GpiEndPath
- GpiFullArc
- GpiGetData
- GpiLabel
- GpiLine
- GpiMarker
- GpiMove
- GpiOffsetElementPointer
- GpiPartialArc
- GpiPointArc
- GpiPolyFillet
- GpiPolyFilletSharp
- GpiPolyLine
- GpiPolyMarker
- GpiPolySpline
- GpiPop
- GpiPutData
- GpiQueryArcParams
- GpiQueryAttrMode
- GpiQueryCurrentPosition
- GpiSetArcParams
- GpiSetAttrMode
- GpiSetAttrs
- GpiSetCharAngle
- GpiSetCharBox
- GpiSetCharDirection
- GpiSetCharMode
- GpiSetCharSet
- GpiSetCharShear
- GpiSetColor
- GpiSetCp
- GpiSetCurrentPosition
- GpiSetEditMode
- GpiSetLineEnd
- GpiSetLineJoin
- GpiSetLineType
- GpiSetLineWidth
- GpiSetMarker
- GpiSetMarkerBox
- GpiSetMarkerSet
- GpiSetMix
- GpiSetModelTransformMatrix

Warning. Some GPI functions cannot be used while an open path definition exists. In particular, you cannot include image or area primitives in a path definition.

Path Operations

The operations performed on a path determine which of the path attributes become applicable. Unlike specifying area brackets, the variety of operations also calls for separate functions instead of specifying options with GpiBeginPath. Specifically, the following six operations can be performed on a path:

- Outline
- Fill
- Modify
- Stroke
- Convert to clip path
- Convert to region

There are other graphics operations provided by the PM -for example, transformations- that can apply to entities other than paths.

Upon completion, GpiFillPath, GpiOutlinePath, GpiStrokePath, and GpiPathToRegion delete the current path definition. This path is not available for any subsequent operations.

GpiModifyPath does not delete the current path definition. An application can modify a path before using one of the aforementioned functions.

The first time your application calls GpiSetClipPath, the current path definition is converted into the current clip path. GpiSetClipPath deletes the path definition upon completion but not the clip path definition.

A path definition can be stored in a graphics segment and, if that segment is retained, the path can be re-created as required. Segments are discussed in Creating and Drawing Retained Graphics.

Path Outline

To draw the outline of the current path, use GpiOutlinePath. This function accepts a path identifier (which must be 1) as input. GpiOutlinePath does not create a geometric line; therefore, the cosmetic line attributes are used.

GpiOutlinePath normally has the same effect as though the lines and arcs used to create the path were drawn without actually being part of the path. Any open figures within the path are not closed automatically. If the path contains character strings that use an outline font, the characters are not filled.

The following functional sequence is recommended for drawing outline text or outline figures:

Function Name	Effect
GpiBeginPath	Starts the path definition.
GpiMove	Moves to the starting point of the first figure.
GpiCharString, GpiFullArc	Uses an outline font to specify text or define the lines in a figure.
GpiEndPath	Ends the path.
GpiOutlinePath	Draws the text or figure outlines.

If you are drawing a complex path, consisting of several character strings that use an outline font, it can take quite some time to produce. If you want to create a only rough draft, you might find it quicker to draw an outline instead of the filled path. This improves performance and

might be acceptable for small characters. When they are small enough, hollow characters can look like filled characters.

You can create hollow characters in either of two ways:

- Outside a path, call GpiCreateLogFont.

 This enables you to define an outline font that uses hollow characters.

- Inside a path, draw the characters, then display the path with GpiOutlinePath.

 Only the character outlines contribute to the path definition even if the font is not hollow. (This might cause ambiguous results.)

Using outlined paths to create character outlines also can be used to define a clip path in the shape of a letter or word.

Path Fill

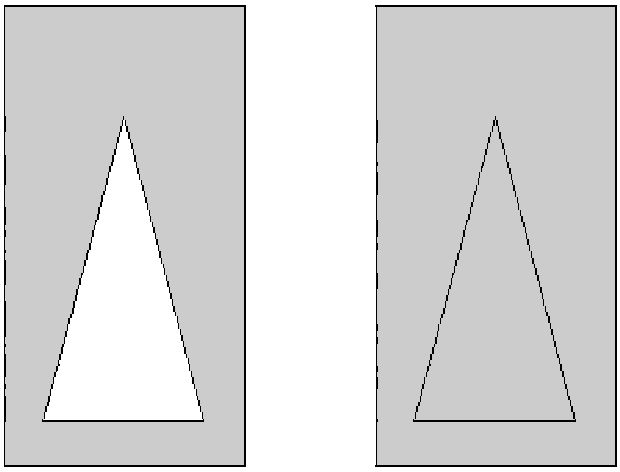
To draw the current path and fill its interior using the current pattern symbol, use GpiFillPath. The path is deleted after the interior is filled. The boundary lines are considered part of the path interior and are drawn with this function. Any open figures in the path definition are closed automatically by GpiFillPath.

This function accepts the path identifier (which must be 1) as input, and either of two construction options:

- FPATH_ALTERNATE (the default)
- FPATH_WINDING (must be selected if the path has been modified).

Paths, like area primitives, can be filled in alternate mode or winding mode. If the path consists of multiple, intersecting figures, the path-fill mode affects the final appearance of the path.

The following figure shows two identical paths that were filled, each using one of the two modes. Each path consists of a triangle drawn within a rectangle. The path on the left was filled using the alternate mode. The path on the right was filled using the winding mode.



Alternate and Winding Fill Modes

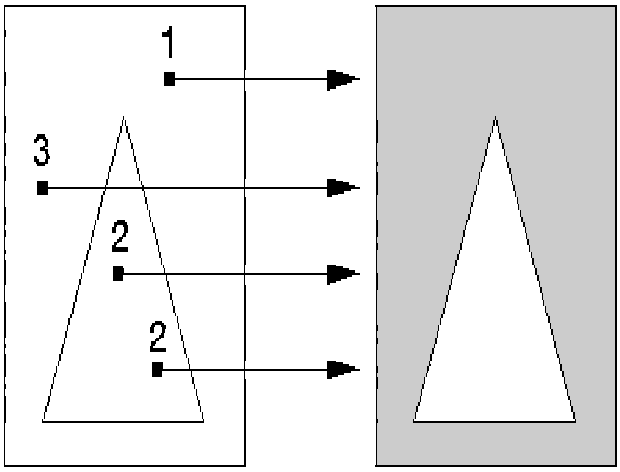
Paths in Alternate Mode

For paths drawn in alternate mode (as with area primitives), the following is true:

- If you have to cross an odd number of boundaries in the path when drawing an imaginary line parallel to the x-axis, a point is included in the filled path from that point to positive infinity.

- If you have to cross an even number of boundaries in the path when drawing an imaginary line parallel to the x-axis, a point is not included in the filled path from that point to positive infinity.

The following figure shows how the operating system determines the filled portion for the path shown in the previous figure. The path outside the triangle, but inside the rectangle, is filled, because the imaginary lines drawn from those points in the positive x-direction intersect the path boundaries an odd number of times.



Calculating Filled Paths Constructed in Alternate Mode

Parts of the path with odd tallys are filled; parts with even tallys are not filled.

Paths in Winding Mode

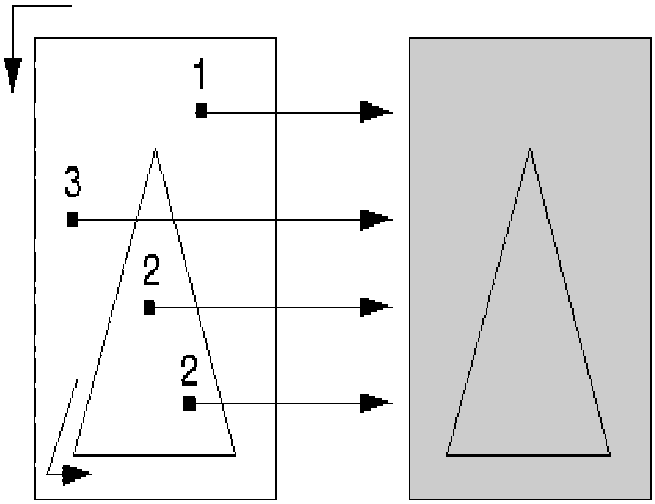
For paths drawn in winding mode (also like area primitives), the direction the boundaries in the path are drawn in determines whether a given point is included in the filled path. The drawn direction of a boundary line depends on both the graphics functions used to draw it and the world coordinates that define it.

For box primitives, for example, if the specified diagonal corner is to the right, to the top, or both, of the current position, the box is drawn counterclockwise. If the specified diagonal corner is to the left, to the bottom, but not both, the box is drawn clockwise.

For a polyline primitive, the direction of the boundary depends on the relative values of the start and end points of each line. For a polygon primitive, the direction of the boundary depends on the relative values of the specified vertices.

To determine whether a given point is included in the filled path, count the number of boundary lines to be crossed to move from that point to infinity. For each boundary line drawn in one direction add one to the tally. For each boundary line drawn in the opposite direction, subtract one from the tally. A point is within the area if the result is nonzero. As with the alternate mode, the imaginary determining line is drawn in the positive x-direction from the point in question toward infinity.

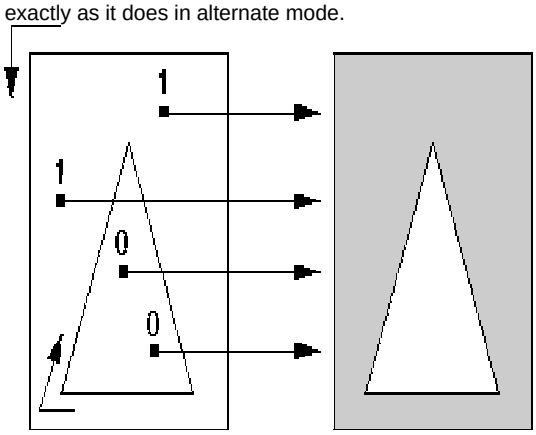
The following figure shows how the operating system determines the filled portion for the path shown in the figure before the previous one. Assume that both the rectangle and triangle were drawn in the same direction, whether clockwise or counterclockwise is immaterial. Both figures are filled, because the number of times the imaginary lines drawn from those points in the positive x-direction intersect the path boundaries is continuously summed. There is never a subtraction of a boundary tally to reduce the total to 0.



Paths Constructed in the Same Direction in Winding Mode

Parts of the path with nonzero tallys are filled; parts with zero tallys are not filled.

If two figures, for example in the following figure, are drawn in different directions, the tally for the inner triangle is 0 and the area looks exactly as it does in alternate mode.



Paths Constructed in Different Directions in Winding Mode

Parts of the path with nonzero tallys are filled; parts with zero tallys are not filled.

Path Modification

To convert the current path to one that describes the envelope of a geometric wide line, use GpiModifyPath. The current geometric line attributes are then applied to the figures within the path. The line end, line join, and geometric line width attributes all must be set before modify-path or stroke-path operations begin, because it is during those times that the attributes are applied. Cosmetic line attributes of width and type have no effect on geometric lines.

GpiModifyPath accepts the path identifier (which must be 1) and the modification option MPATH_STROKE as input.

Subsequently, the modified path can be filled to display the geometric wide line, but only in winding mode. Alternatively, the modified path can be converted into a clip path, but again, only in winding mode.

Open figures within the path are not closed automatically. Figures not explicitly closed with GpiCloseFigure are drawn with line ends rather than line joins at their start and end points. If the figures within the path overlap, the geometric width envelope compensates so that the

overlap portion is not drawn blank in XOR or exclusive-OR mode.

Now the geometric lines can be scaled.

Note: The current transforms are applied to the primitives inside a path bracket when the path is defined. This binds the path definition in device coordinates at that time. The path is unaffected by subsequent transformations, except for those (such as scaling) that affect the *width* of geometric (wide) lines. Since the geometric line width attribute is not applied until the path is converted into a wide line by GpiModifyPath or GpiStrokePath, the width of geometric lines is unaffected by earlier transformations directed at the path definition.

After creating a path bracket, geometric wide lines can be constructed by either:

- Stroking the path (GpiStrokePath),
- Modifying the path (GpiModifyPath), then filling the path (GpiFillPath).

GpiStrokePath is slightly more efficient, but GpiModifyPath and GpiFillPath offer more flexibility (by way of the fill options). After drawing the lines no alterations can be made.

Path-Stroking

To modify and fill the current path in a single operation, use GpiStrokePath. Then the path is drawn immediately to the output device, with the current geometric line attributes applied to the figures within the path. Certain device drivers can use this function to optimize storage.

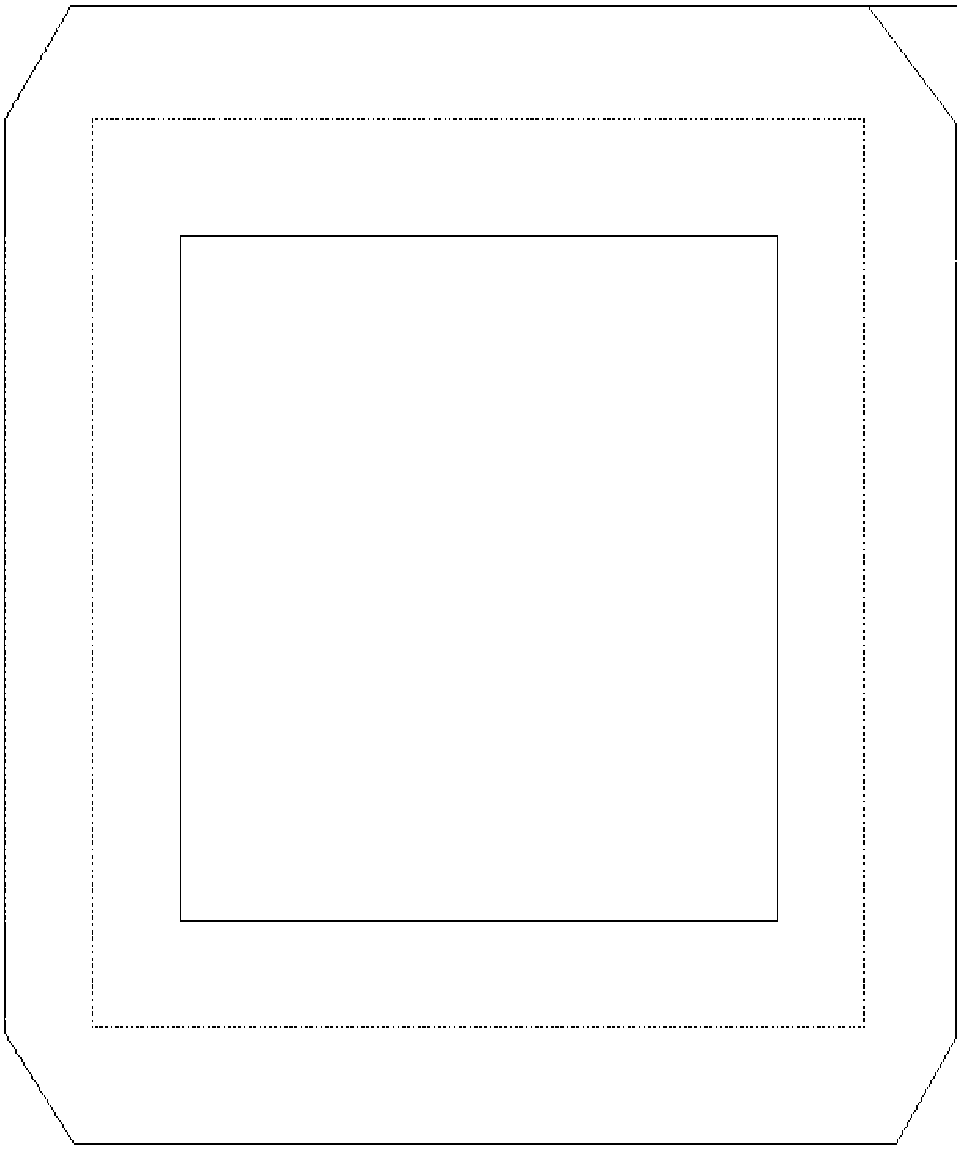
GpiStrokePath automatically fills the path in winding mode with the current area pattern symbol. When GpiStrokePath is complete, the path definition is deleted from the device context.

GpiStrokePath is subject to the same constraints as functions that perform simple line modification:

- The line end, line join, and geometric line width attributes all must be set before stroke operations.
- Cosmetic line attributes of width and type have no effect.
- Any open figures within the path are not closed automatically.
- If figures are not explicitly closed with GpiCloseFigure, they are drawn with line ends rather than line joins at their start and end points.
- If the figures within the path overlap, the geometric width envelope compensates so that the overlap portion is not drawn blank when drawn in XOR (exclusive-OR) mode.

This function accepts as input both the path identifier (which must be 1) and the stroke option (which must be 0L).

When the operating system strokes a path, it draws a geometric line of specified width along the original figure that defined the path and then fills the wide line. If the original figure is not a closed shape, the operating system does not automatically close it before filling the path. The following figure shows the effects of GpiStrokePath on a box originally defined with normal (cosmetic) lines.



Defining Lines with a Geometric Width

The broken line is the figure defined within the path. The solid lines show the path after it has been converted. Each line has a geometric width of *n* coordinate units, and the line joins have been defined as beveled.

After setting these attributes, the application can draw the line with GpiStrokePath. After the lines are drawn, an application can scale stroked paths with a scaling transformation.

As an alternative to the GpiStrokePath, you can convert the path using GpiModifyPath, which does not draw the path on the current output device. To draw a modified path, use GpiFillPath, which draws the path and fills it with the current area-fill pattern in winding mode. A modified path cannot be filled in alternate mode.

On some devices, it might be that the GpiStrokePath method works better than GpiModifyPath, followed by GpiFillPath.

Path Conversion to Clip Path

Clipping is the process an application uses to limit graphics output to a specific area (called the *clipping area*) of the display or page.

There are several clipping functions provided by the PM. However, if your application requires an irregular complex shape for a clipping

area, it must define the shape with a path. To convert the path into a clipping boundary, use GpiSetClipPath. The clip path, as defined by this operation, becomes the current clip path for all subsequent drawing.

This function accepts a path identifier and one of two construction options as input:

- SCP_ALTERNATE (default)
- SCP_WINDING (must be selected if the path has been modified)

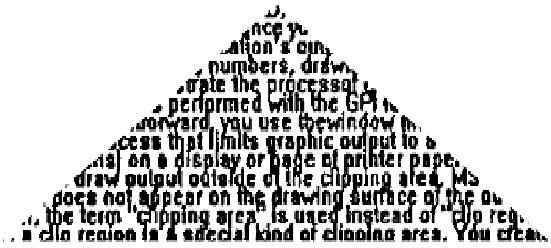
Unlike the path operations described previously, GpiSetClipPath accepts two different path identifiers:

- 1
- 0 (default).

The default path identifier of 0 (SCP_RESET) resets the clip path to infinity, which displays the picture without clipping. If this value is selected, the current clip path definition is discarded instead of stored.

For GpiSetClipPath, a path identifier of 1 (SCP_AND) causes the clip path to be redefined as the mathematical intersection of the stored clip path and the current path definition. For all other path operations, an identifier of 1 specifies the current path as the recipient of the operation. The only method of specifying the clip path as the current path, after GpiSetClipPath has been called, is to call GpiSetClipPath again: the first call with a path identifier of 0; the second, with a path identifier of 1. The path identifiers and the construction mode can be ORed together for certain effects.

Any open figures within a path are closed automatically. The boundaries of the path are considered part of the interior, so any point on the boundary is not clipped. The following figure shows the result of clipping text with a triangular clip path.



Triangular Clip Path

Path Conversion to Region

To convert the current path to a region, use GpiPathToRegion. Then the new region can be modified using any of the region functions that allow the creation of irregular shaped regions.

Any open figures in the path definition are closed automatically by GpiPathToRegion. Upon conversion, the current path definition is discarded just like it is when being converted to a clip path. When GpiPathToRegion is complete, the path definition is deleted automatically.

This function accepts the path identifier (which must be 1) and one of two construction options as input:

- FPATH_ALTERNATE (default)
- FPATH_WINDING (must be selected if the path has been modified)

Using Paths

You can use path functions to:

- Draw geometric lines and filled polygons
- Draw outline text
- Create a triangular clip path

Drawing a Geometric (Wide) Line

The following figure is an example of how to draw a straight line, 10 units wide with rounded ends.

```
#define INCL_GPIPRIMITIVES
#include <os2.h>
void fncPATH01(void){
    POINTL ptl;
    HPS hps;

    GpiSetLineWidthGeom(hps, 10L);       /* Sets line width to ten             */
    GpiSetLineEnd(hps, LINEEND_ROUND);   /* Sets line end to round             */
    GpiBeginPath(hps, 1L);               /* Begins path                        */
    ptl.x = 7;
    ptl.y = 15;
    GpiMove(hps, &ptl);                  /* Sets current position              */
    ptl.x = 450;
    ptl.y = 15;
    GpiLine(hps, &ptl);                  /* Draws line                         */
    GpiEndPath(hps);                     /* Ends path                          */
    GpiStrokePath(hps, 1L, 0L);          /* Draws geometric line               */
} /* fncPATH01 */
```

Drawing Filled Polygons

The following figure is an example of how to draw an empty triangle within a solid rectangle.

```
#define INCL_GPIPATHS
#include <os2.h>
void fncPATH02(void){
    HPS hps;
    POINTL aptl1[4] = {       /*  Array of points for triangle   */
        50,  50,
        100, 100,
        150, 50,
        50,  50  };

    POINTL aptl2[5] = {       /* Array of points for rectangle   */
 25,  25,
 25, 200,
200, 200,
200,  25,
 25,  25 };

    GpiBeginPath(hps, 1L);               /* Begins path                 */
    GpiMove(hps, aptl1);                 /* Sets current position       */
    GpiPolyLine(hps, 4L, aptl1);         /* Plots points for triangle   */
    GpiMove(hps, aptl2);                 /* Sets current position       */
    GpiPolyLine(hps, 5L, aptl2);         /* Plots points for rectangle  */
    GpiEndPath(hps);                     /* Ends path                   */
    GpiFillPath(hps, 1L, FPATH_ALTERNATE); /* Draws triangle and rectangle */
} /* fncPATH02 */
```

Drawing Outline Text

The following figure shows an example of how to draw text from an outline font with GpiOutlinePath.

```
#define INCL_GPIPRIMITIVES
#include <os2.h>
void fncPATH03(void){
    LONG lcid = 1;                          /* Identifier for font           */
    FATTRS fattrs;                          /* Structure for font attributes */
    SIZEF sizfx;                            /* Structure for character box   */
    POINTL ptl;                             /* Structure for starting point  */
    HPS hpsWin;

    fattrs.usRecordLength = sizeof(FATTRS);
    fattrs.fsSelection = FATTR_SEL_OUTLINE;   /* Specifies outline font        */
    fattrs.lMatch = 0;                        /* System determines font        */
    fattrs.szFacename[0] = 0;
    fattrs.idRegistry = 0;
    fattrs.usCodePage = 0;
    fattrs.lMaxBaselineExt = 0;
    fattrs.lAveCharWidth = 0;
    fattrs.fsType = 0;
    fattrs.fsFontUse = FATTR_FONTUSE_OUTLINE; /* Specifies outline font        */

    GpiCreateLogFont(hpsWin,(PSTR8) NULL, lcid, &fattrs);
    GpiSetCharSet(hpsWin, lcid);              /* Sets logical font             */

    sizfx.cx = MAKEFIXED(30, 0);              /* Width of character box        */
    sizfx.cy = MAKEFIXED(30, 0);              /* Height of character box       */
    GpiSetCharBox(hpsWin, &sizfx)             /* Sets size of character box    */

    GpiBeginPath(hpsWin, 1L);                 /* Begins path                   */
    ptl.x = 100;
    ptl.y = 200;
    GpiMove(hpsWin, &ptl);                    /* Sets current position         */
    GpiCharString(hpsWin, 7L, "Outline");     /* Establishes char. string      */
    GpiEndPath(hpsWin);                       /* Ends path                     */
    GpiOutlinePath(hpsWin, 1L, 0L);           /* Draws character string        */
} /* fncPATH03 */
```

Creating a Triangular Clip Path

The following figure shows an example of how to create a triangular clip path and then write text into it.

```
#define INCL_GPIPATHS
#include <os2.h>
void fncPATH04(void){
    POINTL ptlStart;
    HPS hps;
    LONG i;

    POINTL aptl[4] = {            /* Array of points for triangle     */
           35,  45,
          100, 100,
          200,  45,
```

```
            35,  45 };

    GpiBeginPath(hps, 1L);          /* Begins path bracket             */
    GpiMove(hps, aptl);             /* Sets current position           */
    GpiPolyLine(hps, 4L, aptl);     /* Plots points for triangle       */
    GpiEndPath(hps)                 /* Ends path bracket               */

    GpiSetClipPath(hps, 1L, SCP_ALTERNATE | SCP_AND);

    /* Write a block of text.                                          */

    ptlStart.x = 50;
    for (i = 50L; i < 110L; i += 10L) {
        ptlStart.y = i;
        GpiCharStringAt(hps, &ptlStart, 18L, "String of text!");
    } /* for */
} /* fncPATH04 */
```

Regions

A *region* is a graphics object usually composed of one or more rectangles. Converted to clip regions, they are used mainly to define a clipping boundary, in device coordinates, for multiple intersecting rectangles. Clip regions provide the required clipping in an update region during WM_PAINT processing when it is necessary to repaint part of a window. Regions also can be used for area fill.

The following topics are related to the information in this chapter:

- Presentation spaces and device contexts
- Area primitives
- Color and mix modes
- Coordinate spaces and transformations
- Clipping

About Regions

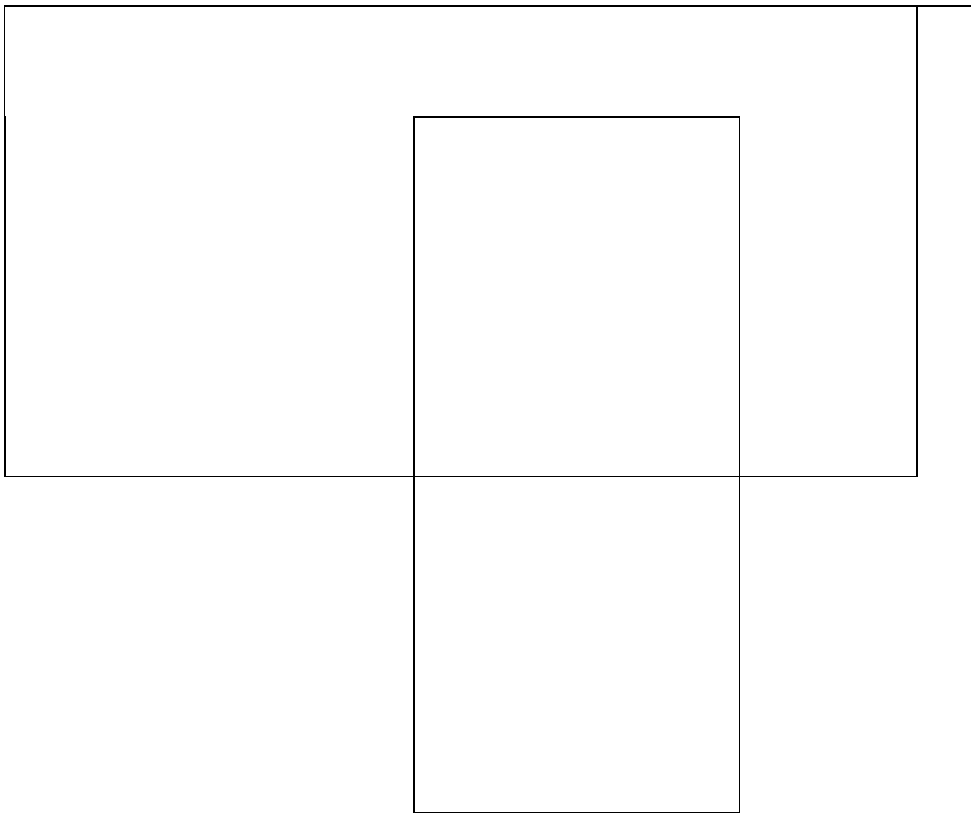
A region consists of one or more overlapping or separate rectangles in an application's device space. The sides of the rectangles are parallel to the x- and y-axes in the device coordinate space. An irregular, nonrectangular path can be converted into a region. However, unless otherwise specified, the assumption of this chapter is that a region is rectangular.

If a region consists of intersecting rectangles, the intersecting sides are always perpendicular. If the rectangles have no common elements, the region is defined as separate rectangles.

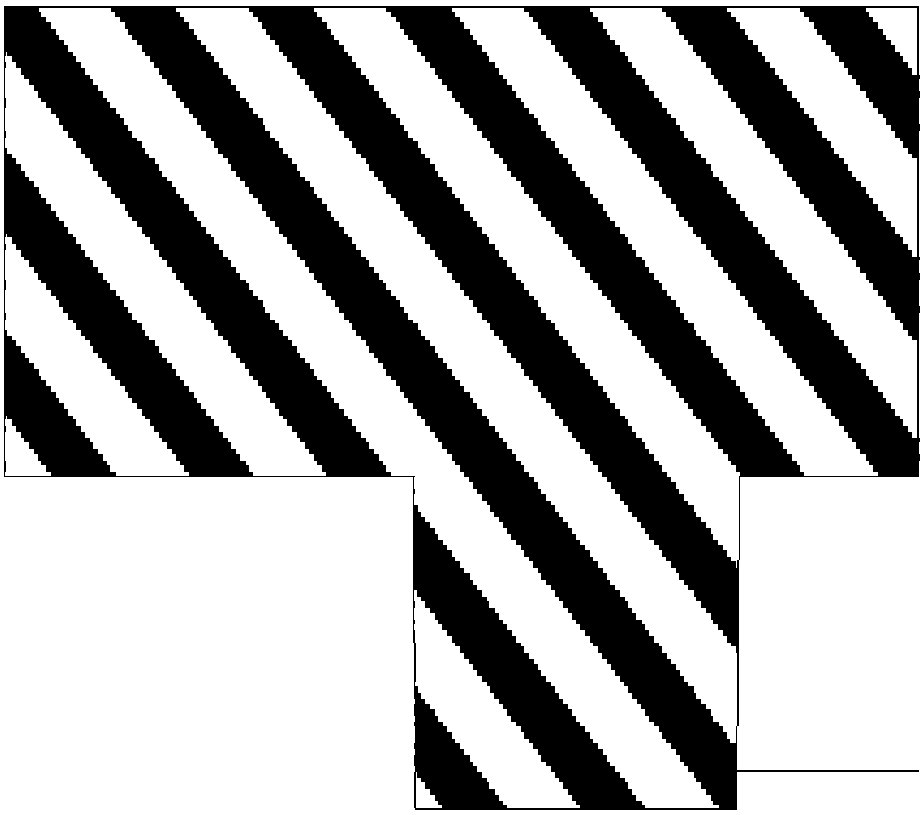
Unlike areas and paths, which are defined in world coordinates, regions are device-dependent and, so, are defined in device coordinates. The device coordinates are *inclusive* (inside the rectangle) at the bottom and left coordinate boundaries, and *exclusive* (outside the rectangle and clipped) at the top and right boundaries.

Each region is created for the device that currently is associated with a presentation space. Also, after creation, regions are available for various operations; a *region handle* identifies the region for subsequent operations.

The following figure shows a region that consists of two intersecting rectangles. An application defined the region by passing an array containing the coordinates for the two rectangles to GpiCreateRegion. The application then drew the region with GpiPaintRegion.



Source Rectangles



Defining a Region

This region comprises two overlapping rectangles.

System Implementation

Regions are device-dependent objects and, therefore, are associated with a particular device. For this reason, when an application specifies a presentation space as input to one of the region functions, that presentation space must be associated with a device context. The associated device context serves to identify the device. Because a region is specific to the device for which it was created, a region should not be created for one device and then used for another.

If the device is a printer, a metafile can be used to store region functions as *escapes* , and the resulting metafile can be displayed on any device. The appearance of the result of any region function is affected by differences in the pel:aspect ratio and device resolution on different printers.

Most of the region-oriented functions use the RECTL structure to define the device-space coordinates of the rectangle. When an application creates a rectangle in a device space and passes its coordinates to a region function, the operating system excludes the top and rightmost edges of the rectangle. This means that an application must add 1 to the values in the *xRight* and *yTop* fields of the RECTL structure to obtain the desired dimensions. For example, if an application requires a region that measures 100-by-100 device units, with a lower-left corner at (10,10), *xRight* and *yTop* should be set to 111 instead of 110.

Region Attributes

Regions do not have a distinct ...BUNDLE data structure associated with them. Unlike primitives or even paths, the purpose of regions is to provide a definition for an operation-not visual output. However, there are two functions that present the region as a visible entity.

GpiPaintRegion paints a region in the presentation space. This function accepts only the region handle as input. GpiPaintRegion does not cause graphics orders to be added to the current segment. (Graphics orders are the smallest, most complete portions of a segment.) Therefore, in retain or draw-and-retain modes, you are advised to use paths or areas rather than regions to ensure the desired effect.

GpiFrameRegion draws a frame around a region by tracing the inner perimeter with a rectangle of a specified size. This function accepts the region handle and the desired frame thickness as input.

A region's visible output is controlled by the following AREABUNDLE attributes:

- Pattern symbol
- Pattern set
- Pattern reference point
- Area foreground color
- Area background color
- Area foreground mix

These attributes follow the current definitions and appear just as area primitives do. The operating system uses the pattern symbol to fill the interior of the region. Any alterations to the fill pattern or reference point affects the appearance of a region, just as it does in an area.

Because neither the painted region nor the region frame have boundary lines, no LINEBUNDLE attributes influence the appearance of a region resulting from GpiPaintRegion or GpiFrameRegion.

Region Creation

Areas and paths are created and defined within a bracket. Regions are defined by a single function, GpiCreateRegion. As input to GpiCreateRegion, supply the total number of, and the coordinates for, each rectangle that contributes to the region. The following applies to region coordinates:

- They define the bottom left and top right coordinates of each rectangle that makes up the region.

- They must be in the range -32768 through +32767.
- They always are device coordinates, because regions always are created and drawn in an application's device space.

The output from GpiCreateRegion is the *region handle* , which identifies the region for subsequent operations. Because each region can be distinctly identified, the region functions allow applications to work with more than one region at a time.

For example, an application can combine two regions with GpiCombineRegion or compare two regions with GpiEqualRegion. GpiQueryRegionRects retrieves the coordinates of the rectangles that make up a region, enabling an application to create a copy of that region. To create the copy, you supply, as input to a new GpiCreateRegion, the exact rectangles returned from GpiQueryRegionRects.

Region Operations

The following operations can be performed on a region:

- Creating Regions
- Drawing Regions
- Moving Regions
- Determining Region Characteristics
- Converting a Path to a Region
- Converting a Region to a Clip Region
- Deleting Regions

Creating Regions

To create a rectangular region, call GpiCreateRegion. This function accepts, as input, the number of rectangles to be ORed into a single region and the coordinates of those rectangles. If the number of rectangles is 0, an empty region is created.

To create a new region from regions that already exist, call GpiCombineRegion. This function accepts, as input, a region handle for the target region, a region handle for each of the two source regions, and an options flag that specifies the way the two source regions are combined.

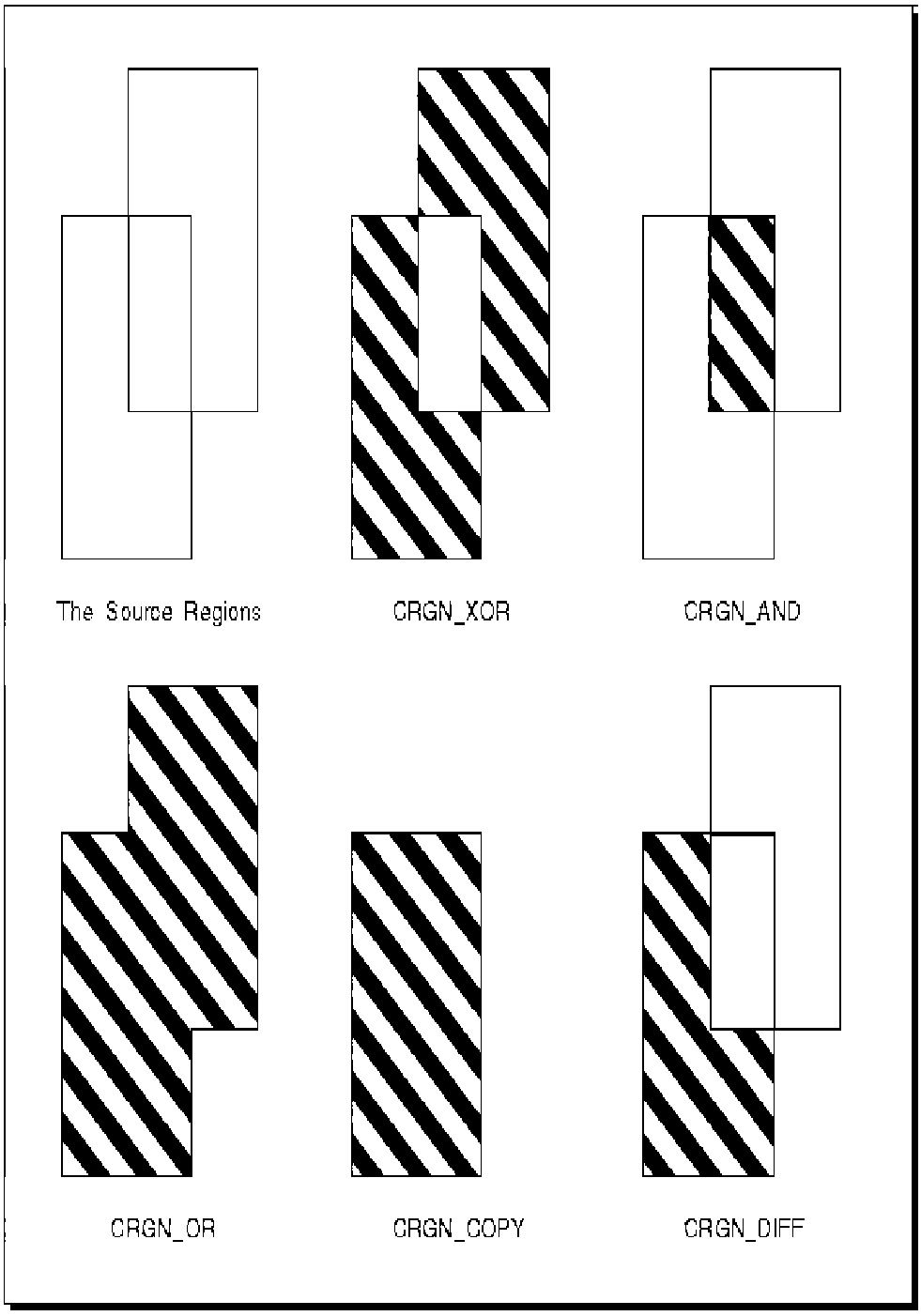
Note: The destination region can be either of the two source regions, in which case, that region is replaced by the new region.

The rectangles are differentiated by one region's being distinctly "the first source rectangle" and the other region labeled "the second source rectangle". The order of the source rectangles affects the way GpiCombineRegion combines them. You can use GpiCombineRegion an indefinite number of times to form complex polygons. However, the function is limited to combining only two regions each time it is called. The two source regions must be created for the same type of device.

Following are the five ways in which two regions can be combined:

Option	Result
CRGN_OR	Union of the two source regions
CRGN_XOR	Symmetric difference of the two source regions
CRGN_COPY	Source region 1 only; source region 2 is ignored
CRGN_AND	Intersection of the two source regions
CRGN_DIFF	Source region 1 minus source region 2

The effects of these different combining operations on overlapping regions are shown in The effects of these different combining operations on overlapping regions are shown in the following figure. Their effects on separate regions are shown in Their effects on separate regions are shown in the figure after the following figure.

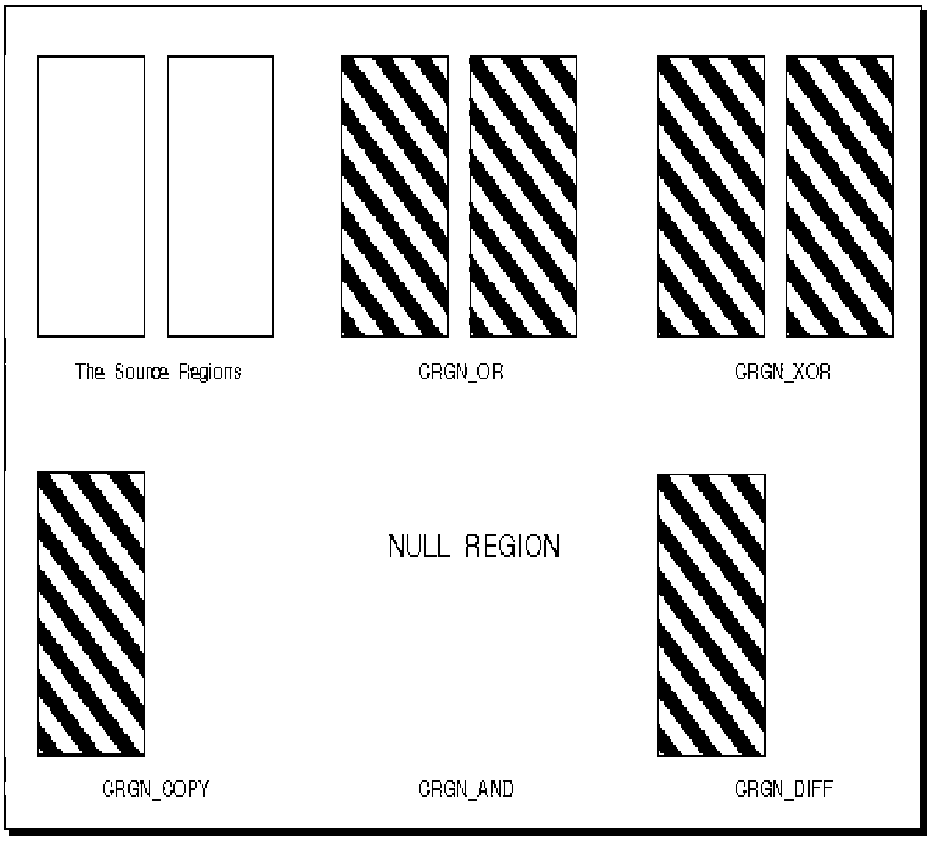


Combining Overlapping Regions

The source regions are two overlapping rectangles. The regions that result from the various combine operations are shown as shaded areas.

Output from GpiCombineRegion tells you whether the resulting region is a NULL region, rectangular region, or complex region. A *complex region* is any region defined by two or more rectangles.

To change the definition of a region while keeping the same region handle, call GpiSetRegion. This function accepts, as input, the number and dimensions of the rectangles that now define the region, just as GpiCreateRegion does. However, GpiSetRegion also accepts the handle of the region to be updated. GpiSetRegion, like GpiCreateRegion, takes a region definition to be the union of a sequence of rectangles (that are effectively ORed together). In the case of GpiCreateRegion, a new region is created and its handle returned. In the case of GpiSetRegion, the previous definition is discarded and replaced by the new definition that is associated with the existing region handle.



Combining Disjoint Regions

The two source regions do not overlap. The regions that result from the various combine operations are shown as shaded areas. The result from a CRGN_AND combining of separate regions is a NULL region.

Moving a Region

To move a region, call GpiOffsetRegion. This function accepts, as input, the region handle and an *offset* value that is added to every coordinate point of the region.

GpiOffsetRegion moves a region by a specific number of device coordinates. By specifying positive or negative *x*- and *y*-values an application can move a region in any direction, relative to its current position.

Determining Region Characteristics

Regions are used in many operations. There are several functions that determine the characteristics of regions because these characteristics influence the outcome of the operation being performed.

To determine whether two regions are identical, call GpiEqualRegion. This function accepts, as input, the two region handles. Regions are identical if the only difference between them is an empty region. For example, a rectangular region whose lower-left corner is at 10,10 is not identical to a region whose lower-left corner is at 50,50, even if the regions have exactly the same dimensions. The two regions also must be of the same device class to be considered identical.

To determine the rectangles that compose a region, call GpiQueryRegionRects, which retrieves the coordinates of a series of rectangles that, when ORed together, define the shape of the regions.

Note: The individual regions returned may differ in size and number from those originally used to specify the region.

GpiQueryRegionRects accepts, as input, the region handle and the maximum number of rectangles that can be returned. The division of the region into multiple rectangles is handled automatically by the PM.

It is not necessary for an application to retrieve all the rectangles in a single function. Your application can call GpiQueryRegionRects any number of times. Therefore, the application must specify the maximum number of rectangles and the rectangle number to start from in each function.

No assumptions can be made about the number and coordinates of rectangles returned except that they will define the specified region precisely. The purpose of specifying the number of rectangles in the query is to ensure that the system does not return more coordinates than can fit in the RECTL array.

To determine whether a point is inside the borders of a region, call GpiPtInRegion. This function is especially useful in applications that must determine whether the mouse pointer is over a region. The particular point must be expressed in device coordinates.

To determine whether any part of a specified rectangle lies within a region, call GpiRectInRegion. This function accepts, as input, the region handle and the rectangle as specified in device coordinates.

To determine the coordinates of the smallest rectangle that encloses a region, call GpiQueryRegionBox. This function accepts the region handle as input. The output from this function tells you whether the bounding region is rectangular, NULL, or complex.

Converting a Region to a Clip Region

Clipping is the process an application uses to limit graphics output to a specific area (called the clipping area) of the display or page.

There are several clipping functions provided by the PM. However, if your application requires a clipping boundary in device coordinates, it must define the boundary with a region. To convert the region into a clipping boundary, call GpiSetClipRegion. The clip region, as defined by this operation, becomes the current clip region of all subsequent drawing operations.

GpiSetClipRegion accepts, as input, the region handle. A NULL region handle sets the clip region to infinity, effectively performing no clipping.

Unlike clip paths, the region that is no longer the current clip region is not deleted. It retains the effects of any changes made to it while it was a clip region, and it can be used with the other region functions, including being reselected as the clip region with GpiSetClipRegion.

You do not have to deselect the current clip region before selecting another. Each selected clip region automatically replaces the one before it. If there is an existing clip region when you call GpiSetClipRegion, it reverts to a normal region, and its handle is returned.

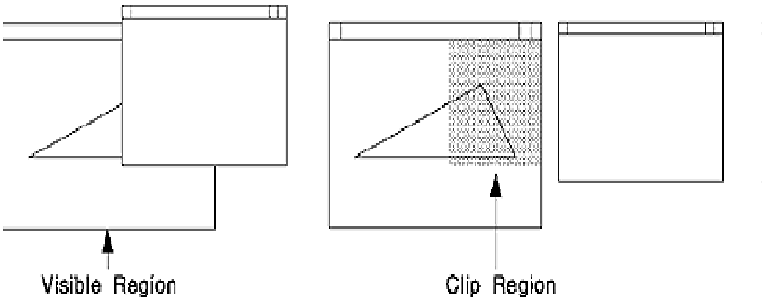
When you have selected the current clip region, none of the region functions described thus far can be used for that region. The following functions can be used with the current clip region:

- GpiQueryClipBox
- GpiIntersectClipRectangle
- GpiExcludeClipRectangle
- GpiOffsetClipRegion
- GpiPtVisible
- GpiRectVisible

These functions are described in Clipping and Boundary Determination. All of these functions work in world coordinates, rather than device coordinates, and therefore, are subject to current transformations.

GpiPtVisible and GpiRectVisible do not apply exclusively to clip regions.

When the screen contents are altered (for example, when a window is sized), you have to be able to repair the part of the screen image affected by the change. The following figure illustrates the necessary region.



Repairing the Screen with Clip Regions

To improve performance of the drawing operation, you can restrict the redrawing and repair work to the affected parts of the screen.

Use WinQueryUpdateRegion to determine whether graphics objects are totally outside the update region and need not be drawn at all. Graphics objects that are within, or are partly outside, the update region should be drawn, and the system will perform the required clipping automatically.

Define a clipping region using the dimensions of the update region. Then call an appropriate GPI drawing request, such as GpiDrawChain, to redraw the screen contents. Any drawing that would occur outside the clip region is discarded according to the standard clipping rules. Only those graphics within the clip region are redrawn.

Deleting a Region

To delete a region, call GpiDestroyRegion. This function accepts, as input, the region handle. If the region is the current clip region, it cannot be deleted. To delete a current clip region, first call GpiSetClipRegion and specify a NULL region handle. The region is no longer the current clip region and can be deleted.

Using Regions

You can use the region functions to:

- Create or delete a region
- Combine regions
- Compare regions
- Move a region
- Paint and frame a region
- Locate a point with respect to a region
- Determine the coordinates of region rectangles

Creating and Deleting a Region

To create a region:

1. Create an array of RECTL structures containing the dimensions of the rectangles that will compose the region.

2. Call GpiCreateRegion to create the region. (This function returns a handle that identifies the region.)

To delete a region, pass the handle returned by the GpiCreateRegion function to GpiDestroyRegion.

The following figure shows how to create and delete a region.

```
#include <os2.h>
void fncREGN01(void){
    HPS  hps;                          /* Presentation-space handle    */
    HRGN hrgn;                         /* Region handle                */
    RECTL arcl[] = {
         25,  50,                      /* Rectangle 1                  */
         75, 100,
         50,  75,                      /* Rectangle 2                  */
        100, 150,
         75, 125,                      /* Rectangle 3                  */
        200, 175,
        150,  75,                      /* Rectangle 4                  */
        250, 150 };

    hrgn = GpiCreateRegion(hps,        /* Creates region               */
        sizeof(arcl) / sizeof (RECTL), /* Number of rectangles in region */
        arcl);                         /* Array of rectangle structures  */
        .
        . /* Work with the region here.                                */
        .
    GpiDestroyRegion(hps, hrgn);   /* Destroys region identified by hrgn */
} /* fncREGN01 */
```

Combining Regions

To combine two regions:

1. Create a region that the operating system can use as the destination region when it combines the source regions.
2. Determine which of the five combining methods to use.
3. Call GpiCombineRegion.

The following figure shows how to combine two regions by using the OR operation:

```
#define INCL_GPIREGIONS
#include <os2.h>
void fncREGN02(void){
    HPS  hps;
    HRGN hrgn1, hrgn2, hrgn3;          /* Region handles                   */
    RECTL rcl1, rcl2;

    rcl1.xLeft = 50; rcl1.yBottom = 100;
    rcl1.xRight = 200; rcl1.yTop = 175;
    hrgn1 = GpiCreateRegion(hps, 1L, &rcl1);        /* First source region  */

    rcl2.xLeft = 125; rcl2.yBottom = 150;
    rcl2.xRight = 225; rcl2.yTop = 200;
    hrgn2 = GpiCreateRegion(hps, 1L, &rcl2);        /* Second source region */

    hrgn3 = GpiCreateRegion(hps, 0L, (PRECTL) NULL); /* Destination region  */

    /* Combine the regions. */
    GpiCombineRegion(hps, hrgn3, hrgn1, hrgn2, CRGN_OR);
} /* fncREGN02 */
```

Comparing Regions

GpiEqualRegion determines whether two regions are identical. The following figure shows how to compare regions.

```
#define INCL_GPIREGIONS
#include <os2.h>
void fncREGN03(void){
    HPS hps;                    /* Presentation-space handle         */
    HRGN hrgn1, hrgn2;          /* Region handles                    */
    LONG lEqual;                /* Return value for GpiEqualRegion   */
    RECTL rcl1, rcl2;           /* Structures for region coordinates */

    rcl1.xLeft = 50; rcl1.yBottom = 100;
    rcl1.xRight = 200; rcl1.yTop = 175;
    hrgn1 = GpiCreateRegion(hps, 1, &rcl1);    /* Creates first region  */

    rcl2.xLeft = 125; rcl2.yBottom = 150;
    rcl2.xRight = 225; rcl2.yTop = 200;
    hrgn2 = GpiCreateRegion(hps, 1, &rcl2);    /* Creates second region */

    lEqual = GpiEqualRegion(hps, hrgn1, hrgn2); /* Compares regions     */
    if (lEqual == EQRGN_EQUAL)
    {
        .
        .   /* Regions are equal.     */
        .
    }
    else if (lEqual == EQRGN_NOTEQUAL)
    {
        .
        .   /* Regions are not equal. */
        .
    }
    else if (lEqual == EQRGN_ERROR)
    {
        .
        .   /* An error occurred.     */
        .
    }
} /* fncREGN03 */
```

Offsetting a Region

GpiOffsetRegion moves a region, by a specified offset, from its current position (in world space). This function must be called with the address of a POINTL structure that contains an x and a y translation factor. The following figure shows how to offset a region.

```
} /* fncREGN04 */
    POINTL ptlNewPos;            /* Structure for offset value  */

    ptlNewPos.x = 200;           /* Sets x offset               */
    ptlNewPos.y = 10;            /* Sets y offset               */

    GpiOffsetRegion(hps, hrgn, &ptlNewPos);    /* Offsets region */
```

Painting a Region

GpiPaintRegion fills a region with the current fill pattern, using the colors and mix mode that appear in the current AREABUNDLE structure. GpiFrameRegion draws a frame around a region by tracing the perimeter of the region with a rectangle of a specified size. The following figure shows how to change the fill-pattern color to green, paint a region, and draw a frame around the region.

```
    SIZEL sizl;                       /* Structure for size of frame */
    HPS  hps;                         /* Presentation-space handle   */
    HRGN hrgn;                        /* Region handle               */

    GpiSetColor(hps, CLR_DARKPINK);
    GpiPaintRegion(hps, hrgn);        /* Paint region dark pink      */
    GpiSetColor(hps, CLR_BLACK);
    sizl.cx = 5;
    sizl.cy = 5;
    GpiFrameRegion(hps, hrgn, &sizl);   /* 5-by-5 black frame        */
```

Locating a Point with Respect to a Region

To determine whether the mouse pointer lies within a region:

1. Retrieve the mouse-pointer coordinates and store them in a POINTL structure.
2. Call GpiPtInRegion, passing it a handle that identifies the appropriate region and the address of the POINTL structure from Step 1.
3. Examine the value that GpiPtInRegion returns to determine whether the point lies within the region.

The following figure shows how to locate the mouse pointer with respect to a region.

```
#define INCL_GPIREGIONS
#include <os2.h>
void fncREGN04(void){
    POINTL ptlCurPos;      /* Mouse coordinates              */
    HPS hps;               /* Presentation-space handle      */
    HRGN hrgn;             /* Region handle                  */
    LONG lPosition;        /* Return value for GpiPtInRegion */

    /* Determine mouse coordinates and store in ptlCurPos.   */
    lPosition = GpiPtInRegion(hps, hrgn, &ptlCurPos);
    if (lPosition == PRGN_INSIDE) {
        .
        . /* Point lies within region.  */
        .
    } /* if */
    else if (lPosition == PRGN_OUTSIDE) {
        .
        . /* Point lies outside region. */
        .
    } /* else-if */
```

Determining Coordinates of Rectangles in a Region

If a region consists of more than one rectangle, you can call GpiQueryRegionRects to retrieve the coordinates of the lower-left and upper-right corners of each rectangle.

If you use GpiQueryRegionRects to retrieve every rectangle requested to define a region, the function retrieves the coordinates of as many contiguous rectangles as required. GpiQueryRegionRects returns the coordinates of the rectangles that define a region to an array of

RECTL structures, and returns the number of rectangles that were requested for the definition to the *crcReturned* field of the RGNRECT structure. Your RECTL array may not be large enough to hold all of the rectangles. Specify the maximum it can accept and request the remainder in subsequent functions if necessary.

To determine the coordinates of the rectangles that form a region, follow these steps:

1. Create an array of RECTL structures that will receive the rectangle coordinates. More rectangles may be required to define the region than were required to create it.

2. Fill in the *ircStart*, *crc* , and *usDirection* fields of the RGNRECT structure. The *crc* field can specify more rectangles than will be returned by GpiQueryRegionRects.

3. Call GpiQueryRegionRects to retrieve the coordinates.

The following figure shows how to determine the number of rectangles that define a region, the coordinates of the rectangles in that region, and how to create a new region using those coordinates.

```
#define INCL_GPIREGIONS
#include <os2.h>
void fncREGN05(void){
    RGNRECT rgnrc;              /* Structure for region rectangles       */
    RECTL arcl1[5];             /* Array for determining rectangle count */
    PRECTL parcl;               /* Array for rectangle coordinates       */
    ULONG cRects = 0;           /* Total number of rectangles in region  */
    HPS hps;
    HRGN hrgn3;

    rgnrc.ircStart = 1;         /* Rectangle to start with               */
    rgnrc.crc = 5;              /* Number of rectangles to query         */
    rgnrc.ulDirection = RECTDIR_LFRT_BOTTOP;      /* Direction to query   */

    /*********************************************************************/
    /*  Determine the total number of rectangles in the region by        */
    /*  repeatedly calling GpiQueryRegionRects with an array of 5 RECTL  */
    /*  structures. The loop terminates when the function retrieves less */
    /*  than 5 rectangles.                                               */
    /*********************************************************************/
    do {
        GpiQueryRegionRects(hps, /* Handle of presentation space         */
            hrgn3,               /* Region to query                      */
            (PRECTL) NULL,       /* Gets all rectangles in region        */
            &rgnrc,              /* Structure with rectangle data        */
            arcl1);              /* Array of structures for coordinates  */

        cRects += rgnrc.crcReturned;
    } while (rgnrc.crcReturned == rgnrc.crc); /* While 5 rects retrieved  */

    //cRects = rgnrc.crcReturned + (rgnrc.ircStart - 1);

    rgnrc.ircStart = 0;                 /* Rectangle to start with       */
    rgnrc.crc = cRects;                 /* Number of rectangles to query */

    /* Allocate enough memory for all RECTL structures.                  */
    parcl = (PRECTL) malloc(cRects * sizeof(RECTL));

    /* Fill array with coordinates of all rectangles.                    */
    GpiQueryRegionRects(hps,     /* Handle of presentation space         */
        hrgn3,                   /* Region to query                      */
        (PRECTL) NULL,           /* Gets all rectangles in region        */
        &rgnrc,                  /* Structure with rectangle data        */
        parcl);                  /* Array of structures for coordinates  */
} /* fncREGN05 */
```

Notices

November 1996

The following paragraph does not apply to the United Kingdom or any country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This publication could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time.

It is possible that this publication may contain reference to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming, or services in your country.

Requests for technical information about IBM products should be made to your IBM reseller or IBM marketing representative.

Copyright Notices

COPYRIGHT LICENSE: This publication contains printed sample application programs in source language, which illustrate OS/2 programming techniques. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the OS/2 application programming interface.

Each copy of any portion of these sample programs or any derivative work, which is distributed to others, must include a copyright notice as follows: "(C) (your company name) (year). All rights reserved."

(C) Copyright International Business Machines Corporation 1994,1996. All rights reserved.
Note to U.S. Government Users - Documentation related to restricted rights - Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Disclaimers

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:
IBM
PM
Presentation Manager
OS/2
SAA
System Application Architecture
Workplace Shell

The following terms are trademarks of other companies:

C++	AT&T, Inc.
Helvetica	Linotype

Double-Byte Character Set (DBCS)

Throughout this publication, you will see references to specific value for character strings. The values are for single-byte character set (SBCS). If you use the double-byte character set (DBCS), note that one DBCS character equals two SBCS characters.

Glossary

This glossary defines many of the terms used in this book. It includes terms and definitions from the *IBM Dictionary of Computing* , as well as terms specific to the OS/2 operating system and the Presentation Manager. It is not a complete glossary for the entire OS/2 operating system; nor is it a complete dictionary of computer terms.

Other primary sources for these definitions are:

- The *American National Standard Dictionary for Information Systems* , ANSI X3.172-1990, copyrighted 1990 by the American National Standards Institute, 11 West 42nd Street, New York, New York 10036. These definitions are identified by the symbol (A) after the definition.
- The *Information Technology Vocabulary* , developed by Subcommittee 1, Joint Technical Committee 1, of the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC JTC1/SC1). Definitions of published parts of this vocabulary are identified by the symbol (I) after the definition; definitions taken from draft international standards, committee drafts, and working papers being developed by ISO/IEC JTC1/SC1 are identified by the symbol (T) after the definition, indicating that final agreement has not yet been reached among the participating National Bodies of SC1.

Glossary Listing

Select a starting letter of glossary terms:

A N
B O
C P
D Q
E R
F S
G T
H U
I V
J W
K X
L Y
M Z

Glossary - A

accelerator -In SAA Common User Access architecture, a key or combination of keys that invokes an application-defined function.

accelerator table -A table used to define which key strokes are treated as *accelerators* and the commands they are translated into.

access mode -The manner in which an application gains access to a file it has opened. Examples of access modes are read-only, write-only, and read/write.

access permission -All access rights that a user has regarding an object. (I)

action -One of a set of defined tasks that a computer performs. Users request the application to perform an action in several ways, such as typing a command, pressing a function key, or selecting the action name from an action bar or menu.

action bar -In SAA Common User Access architecture, the area at the top of a window that contains choices that give a user access to actions available in that window.

action point -The current position on the screen at which the pointer is pointing. Contrast with *hot spot* and *input focus* .

active program -A program currently running on the computer. An active program can be interactive (running and receiving input from the user) or noninteractive (running but not receiving input from the user). See also *interactive program* and *noninteractive program* .

active window -The window with which the user is currently interacting.

address space -(1) The range of addresses available to a program. (A) (2) The area of virtual storage available for a particular job.

alphanumeric video output -Output to the logical video buffer when the video adapter is in text mode and the logical video buffer is addressed by an application as a rectangular array of character cells.

American National Standard Code for Information Interchange -The standard code, using a coded character set consisting of 7-bit coded characters (8 bits including parity check), that is used for information interchange among data processing systems, data communication systems, and associated equipment. The ASCII set consists of control characters and graphic characters. (A)

Note: IBM has defined an extension to ASCII code (characters 128-255).

anchor -A window procedure that handles Presentation Manager message conversions between an icon procedure and an application.

anchor block -An area of Presentation-Manager-internal resources to allocated process or thread that calls WinInitialize.

anchor point -A point in a window used by a program designer or by a window manager to position a subsequently appearing window.

ANSI -American National Standards Institute.

APA -All points addressable.

API -Application programming interface.

application -A collection of software components used to perform specific types of work on a computer; for example, a payroll application, an airline reservation application, a network application.

application object -In SAA Advanced Common User Access architecture, a form that an application provides for a user; for example, a spreadsheet form. Contrast with *user object* .

application programming interface (API) -A functional interface supplied by the operating system or by a separately orderable licensed program that allows an application program written in a high-level language to use specific data or functions of the operating system or the licensed program.

application-modal -Pertaining to a message box or dialog box for which processing must be completed before further interaction with any other window owned by the same application may take place.

area -In computer graphics, a filled shape such as a solid rectangle.

ASCII -American National Standard Code for Information Interchange.

ASCIIZ -A string of ASCII characters that is terminated with a byte containing the value 0.

aspect ratio -In computer graphics, the width-to-height ratio of an area, symbol, or shape.

asynchronous (ASYNC) -(1) Pertaining to two or more processes that do not depend upon the occurrence of specific events such as common timing signals. (T) (2) Without regular time relationship; unexpected or unpredictable with respect to the execution of program instructions. See also *synchronous* .

atom -A constant that represents a string. As soon as a string has been defined as an atom, the atom can be used in place of the string to save space. Strings are associated with their respective atoms in an *atom table* . See also *integer atom* .

atom table -A table used to relate *atoms* with the strings that they represent. Also in the table is the mechanism by which the presence of a string can be checked.

atomic operation -An operation that completes its work on an object before another operation can be performed on the same object.

attribute -A characteristic or property that can be controlled, usually to obtain a required appearance; for example, the color of a line. See also *graphics attributes* and *segment attributes* .

automatic link -In Information Presentation Facility (IPF), a link that begins a chain reaction at the primary window. When the user selects the primary window, an automatic link is activated to display secondary windows.

AVIO -Advanced Video Input/Output.

Glossary - B

Bezier curve -(1) A mathematical technique of specifying smooth continous lines and surfaces, which require a starting point and a finishing point with several intermediate points that influence or control the path of the linking curve. Named after Dr. P. Bezier. (2) (D of C) In the AIX Graphics Library, a cubic spline approximation to a set of four control points that passes through the first and fourth control points and that has a continuous slope where two spline segments meet. Named after Dr. P. Bezier.

background -(1) In multiprogramming, the conditions under which low-priority programs are executed. Contrast with *foreground* . (2) An active session that is not currently displayed on the screen.

background color -The color in which the background of a graphic primitive is drawn.

background mix -An attribute that determines how the background of a graphic primitive is combined with the existing color of the graphics presentation space. Contrast with *mix* .

background program -In multiprogramming, a program that executes with a low priority. Contrast with *foreground program* .

bit map -A representation in memory of the data displayed on an APA device, usually the screen.

block -(1) A string of data elements recorded or transmitted as a unit. The elements may be characters, words, or logical records. (T) (2) To record data in a block. (3) A collection of contiguous records recorded as a unit. Blocks are separated by interblock gaps and each block may contain one or more records. (A)

block device -A storage device that performs I/O operations on blocks of data called *sectors* . Data on block devices can be randomly accessed. Block devices are designated by a drive letter (for example, **C:**).

blocking mode -A condition set by an application that determines when its threads might block. For example, an application might set the Pipemode parameter for the DosCreateNPipe function so that its threads perform I/O operations to the named pipe block when no data is available.

border -A visual indication (for example, a separator line or a background color) of the boundaries of a window.

boundary determination -An operation used to compute the size of the smallest rectangle that encloses a graphics object on the screen.

breakpoint -(1) A point in a computer program where execution may be halted. A breakpoint is usually at the beginning of an instruction where halts, caused by external intervention, are convenient for resuming execution. (T) (2) A place in a program, specified by a command or a condition, where the system halts execution and gives control to the workstation user or to a specified program.

broken pipe -When all of the handles that access one end of a pipe have been closed.

bucket -One or more fields in which the result of an operation is kept.

buffer -(1) A portion of storage used to hold input or output data temporarily. (2) To allocate and schedule the use of buffers. (A)

button -A mechanism used to request or initiate an action. See also *barrel buttons* , *bezel buttons* , *mouse button* , *push button* , and *radio button* .

byte pipe -Pipes that handle data as byte streams. All unnamed pipes are byte pipes. Named pipes can be byte pipes or message pipes. See *byte stream* .

byte stream -Data that consists of an unbroken stream of bytes.

Glossary - C

cache -A high-speed buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

cached micro presentation space -A presentation space from a Presentation-Manager-owned store of micro presentation spaces. It can be used for drawing to a window only, and must be returned to the store when the task is complete.

CAD -Computer-Aided Design.

call -(1) The action of bringing a computer program, a routine, or a subroutine into effect, usually by specifying the entry conditions and jumping to an entry point. (I) (A) (2) To transfer control to a procedure, program, routine, or subroutine.

calling sequence -A sequence of instructions together with any associated data necessary to execute a call. (T)

Cancel -An action that removes the current window or menu without processing it, and returns the previous window.

cascaded menu -In the OS/2 operating system, a menu that appears when the arrow to the right of a cascading choice is selected. It contains a set of choices that are related to the cascading choice. Cascaded menus are used to reduce the length of a menu. See also *cascading choice* .

cascading choice -In SAA Common User Access architecture, a choice in a menu that, when selected, produces a cascaded menu containing other choices. An arrow (->) appears to the right of the cascading choice.

CASE statement -In PM programming, provides the body of a window procedure. There is usually one CASE statement for each message type supported by an application.

CGA -Color graphics adapter.

chained list -A list in which the data elements may be dispersed but in which each data element contains information for locating the next. (T) Synonymous with *linked list* .

character -A letter, digit, or other symbol.

character box -In computer graphics, the boundary that defines, in world coordinates, the horizontal and vertical space occupied by a single character from a character set. See also *character mode* . Contrast with *character cell* .

character cell -The physical, rectangular space in which any single character is displayed on a screen or printer device. Position is addressed by row and column coordinates. Contrast with *character box* .

character code -The means of addressing a character in a character set, sometimes called *code point* .

character device -A device that performs I/O operations on one character at a time. Because character devices view data as a stream of bytes, character-device data cannot be randomly accessed. Character devices include the keyboard, mouse, and printer, and are referred to by name.

character mode -A mode that, in conjunction with the font type, determines the extent to which graphics characters are affected by the character box, shear, and angle attributes.

character set -(1) An ordered set of unique representations called characters; for example, the 26 letters of English alphabet, Boolean 0 and 1, the set of symbols in the Morse code, and the 128 ASCII characters. (A) (2) All the valid characters for a programming language or for a computer system. (3) A group of characters used for a specific reason; for example, the set of characters a printer can print.

check box -In SAA Advanced Common User Access architecture, a square box with associated text that represents a choice. When a user selects a choice, an X appears in the check box to indicate that the choice is in effect. The user can clear the check box by selecting the choice again. Contrast with *radio button* .

check mark - (1) (D of C) In SAA Advanced Common User Access architecture, a symbol that shows that a choice is currently in effect. (2) The symbol that is used to indicate a selected item on a pull-down menu.

child process -In the OS/2 operating system, a process started by another process, which is called the parent process. Contrast with *parent process* .

child window -A window that appears within the border of its parent window (either a primary window or another child window). When the parent window is resized, moved, or destroyed, the child window also is resized, moved, or destroyed; however, the child window can be moved or resized independently from the parent window, within the boundaries of the parent window. Contrast with *parent window* .

choice -(1) An option that can be selected. The choice can be presented as text, as a symbol (number or letter), or as an icon (a pictorial symbol). (2) (D of C) In SAA Common User Access architecture, an item that a user can select.

chord -(1) To press more than one button on a pointing device while the pointer is within the limits that the user has specified for the operating environment. (2) (D of C) In graphics, a short line segment whose end points lie on a circle. Chords are a means for producing a circular image from straight lines. The higher the number of chords per circle, the smoother the circular image.

class -In object-oriented design or programming, a group of objects that share a common definition and that therefore share common properties, operations, and behavior. Members of the group are called instances of the class.

class method -In System Object Model, an action that can be performed on a class object. Synonymous with factory method.

class object -In System Object Model, the run-time implementation of a class.

class style -The set of properties that apply to every window in a window class.

client -(1) A functional unit that receives shared services from a server. (T) (2) A user, as in a client process that uses a named pipe or queue that is created and owned by a server process.

client area -The part of the window, inside the border, that is below the menu bar. It is the user's work space, where a user types information and selects choices from selection fields. In primary windows, it is where an application programmer presents the objects that a user works on.

client program -An application that creates and manipulates instances of classes.

client window -The window in which the application displays output and receives input. This window is located inside the frame window, under the window title bar and any menu bar, and within any scroll bars.

clip limits -The area of the paper that can be reached by a printer or plotter.

clipboard -In SAA Common User Access architecture, an area of computer memory, or storage, that temporarily holds data. Data in the clipboard is available to other applications.

clipping -In computer graphics, removing those parts of a display image that lie outside a given boundary. (I) (A)

clipping area -The area in which the window can paint.

clipping path -A clipping boundary in world-coordinate space.

clock tick -The minimum unit of time that the system tracks. If the system timer currently counts at a rate of X Hz, the system tracks the time every 1/X of a second. Also known as *time tick* .

CLOCK$ -Character-device name reserved for the system clock.

code page -An assignment of graphic characters and control-function meanings to all code points.

code point -(1) Synonym for *character code* . (2) (D of C) A 1-byte code representing one of 256 potential characters.

code segment -An executable section of programming code within a load module.

color dithering -See *dithering* .

color graphics adapter (CGA) -An adapter that simultaneously provides four colors and is supported by all IBM Personal Computer and Personal System/2 models.

command -The name and parameters associated with an action that a program can perform.

command area -An area composed of a command field prompt and a command entry field.

command entry field -An entry field in which users type commands.

command line -On a display screen, a display line, sometimes at the bottom of the screen, in which only commands can be entered.

command mode -A state of a system or device in which the user can enter commands.

command prompt -A field prompt showing the location of the command entry field in a panel.

Common Programming Interface (CPI) -Definitions of those application development languages and services that have, or are intended to have, implementations on and a high degree of commonality across the SAA environments. One of the three SAA architectural areas. See also *Common User Access architecture* .

Common User Access (CUA) architecture - Guidelines for the dialog between a human and a workstation or terminal. One of the three

SAA architectural areas. See also *Common Programming Interface* .

compile -To translate a program written in a higher-level programming language into a machine language program.

composite window -A window composed of other windows (such as a frame window, frame-control windows, and a client window) that are kept together as a unit and that interact with each other.

computer-aided design (CAD) -The use of a computer to design or change a product, tool, or machine, such as using a computer for drafting or illustrating.

COM1, COM2, COM3 -Character-device names reserved for serial ports 1 through 3.

CON -Character-device name reserved for the console keyboard and screen.

conditional cascaded menu -A pull-down menu associated with a menu item that has a cascade mini-push button beside it in an object's pop-up menu. The conditional cascaded menu is displayed when the user selects the mini-push button.

container -In SAA Common User Access architecture, an object that holds other objects. A folder is an example of a container object. See also *folder* and *object* .

contextual help -In SAA Common User Access Architecture, help that gives specific information about the item the cursor is on. The help is contextual because it provides information about a specific item as it is currently being used. Contrast with *extended help* .

contiguous -Touching or joining at a common edge or boundary, for example, an unbroken consecutive series of storage locations.

control -In SAA Advanced Common User Access architecture, a component of the user interface that allows a user to select choices or type information; for example, a check box, an entry field, a radio button.

control area -A storage area used by a computer program to hold control information. (I) (A)

Control Panel -In the Presentation Manager, a program used to set up user preferences that act globally across the system.

Control Program -(1) The basic functions of the operating system, including DOS emulation and the support for keyboard, mouse, and video input/output. (2) A computer program designed to schedule and to supervise the execution of programs of a computer system. (I) (A)

control window -A window that is used as part of a composite window to perform simple input and output tasks. Radio buttons and check boxes are examples.

control word -An instruction within a document that identifies its parts or indicates how to format the document.

coordinate space -A two-dimensional set of points used to generate output on a video display of printer.

Copy -A choice that places onto the clipboard, a copy of what the user has selected. See also *Cut* and *Paste* .

correlation -The action of determining which element or object within a picture is at a given position on the display. This follows a *pick* operation.

coverpage window -A window in which the application's help information is displayed.

CPI -Common Programming Interface.

critical extended attribute -An extended attribute that is necessary for the correct operation of the system or a particular application.

critical section -(1) In programming languages, a part of an asynchronous procedure that cannot be executed simultaneously with a certain part of another asynchronous procedure. (I)

Note: Part of the other asynchronous procedure also is a critical section. (2) A section of code that is not reentrant; that is, code that can be executed by only one thread at a time.

CUA architecture -Common User Access architecture.

current position -In computer graphics, the position, in user coordinates, that becomes the starting point for the next graphics routine, if that routine does not explicitly specify a starting point.

cursor -A symbol displayed on the screen and associated with an input device. The cursor indicates where input from the device will be placed. Types of cursors include text cursors, graphics cursors, and selection cursors. Contrast with *pointer* and *input focus* .

Cut -In SAA Common User Access architecture, a choice that removes a selected object, or a part of an object, to the clipboard, usually compressing the space it occupied in a window. See also *Copy* and *Paste* .

Glossary - D

daisy chain -A method of device interconnection for determining interrupt priority by connecting the interrupt sources serially.

data segment -A nonexecutable section of a program module; that is, a section of a program that contains data definitions.

data structure -The syntactic structure of symbolic expressions and their storage-allocation characteristics. (T)

data transfer -The movement of data from one object to another by way of the clipboard or by direct manipulation.

DBCS -Double-byte character set.

DDE -Dynamic data exchange.

deadlock -(1) Unresolved contention for the use of a resource. (2) An error condition in which processing cannot continue because each of two elements of the process is waiting for an action by, or a response from, the other. (3) An impasse that occurs when multiple processes are waiting for the availability of a resource that will not become available because it is being held by another process that is in a similar wait state.

debug -To detect, diagnose, and eliminate errors in programs. (T)

decipoint -In printing, one tenth of a point. There are 72 points in an inch.

default procedure -A function provided by the Presentation Manager Interface that may be used to process standard messages from dialogs or windows.

default value -A value assumed when no value has been specified. Synonymous with assumed value. For example, in the graphics programming interface, the default line-type is 'solid'.

definition list -A type of list that pairs a term and its description.

delta -An application-defined threshold, or number of container items, from either end of the list.

descendant -See *child process* .

descriptive text -Text used in addition to a field prompt to give more information about a field.

Deselect all -A choice that cancels the selection of all of the objects that have been selected in that window.

Desktop Manager -In the Presentation Manager, a window that displays a list of groups of programs, each of which can be started or stopped.

desktop window -The window, corresponding to the physical device, against which all other types of windows are established.

detached process -A background process that runs independent of the parent process.

detent -A point on a slider that represents an exact value to which a user can move the slider arm.

device context -A logical description of a data destination such as memory, metafile, display, printer, or plotter. See also *direct device context* , *information device context* , *memory device context* , *metafile device context* , *queued device context* , and *screen device context* .

device driver -A file that contains the code needed to attach and use a device such as a display, printer, or plotter.

device space -(1) Coordinate space in which graphics are assembled after all GPI transformations have been applied. Device space is defined in device-specific units. (2) (D of C) In computer graphics, a space defined by the complete set of addressable points of a display device. (A)

dialog -The interchange of information between a computer and its user through a sequence of requests by the user and the presentation of responses by the computer.

dialog box -In SAA Advanced Common User Access architecture, a movable window, fixed in size, containing controls that a user uses to provide information required by an application so that it can continue to process a user request. See also *message box, primary window, secondary window* . Also known as a *pop-up window* .

Dialog Box Editor -A *WYSIWYG* editor that creates dialog boxes for communicating with the application user.

dialog item -A component (for example, a menu or a button) of a dialog box. Dialog items are also used when creating dialog templates.

dialog procedure -A dialog window that is controlled by a window procedure. It is responsible for responding to all messages sent to the dialog window.

dialog tag language -A markup language used by the DTL compiler to create dialog objects.

dialog template -The definition of a dialog box, which contains details of its position, appearance, and window ID, and the window ID of each of its child windows.

direct device context -A logical description of a data destination that is a device other than the screen (for example, a printer or plotter), and where the output is not to go through the spooler. Its purpose is to satisfy queries. See also *device context* .

direct manipulation -The user's ability to interact with an object by using the mouse, typically by dragging an object around on the Desktop and dropping it on other objects.

direct memory access (DMA) -A technique for moving data directly between main storage and peripheral equipment without requiring processing of the data by the processing unit.(T)

directory -A type of file containing the names and controlling information for other files or other directories.

display point -Synonym for *pel* .

dithering -(1) The process used in color displays whereby every other pel is set to one color, and the intermediate pels are set to another. Together they produce the effect of a third color at normal viewing distances. This process can only be used on solid areas of color; it does not work, for example, on narrow lines. (2) (D of C) In computer graphics, a technique of interleaving dark and light pixels so that the resulting image looks smoothly shaded when viewed from a distance.

DMA -Direct memory access.

DOS Protect Mode Interface (DPMI) -An interface between protect mode and real mode programs.

double-byte character set (DBCS) -A set of characters in which each character is represented by two bytes. Languages such as Japanese, Chinese, and Korean, which contain more characters than can be represented by 256 code points, require double-byte character sets. Since each character requires two bytes, the entering, displaying, and printing of DBCS characters requires hardware and software that can support DBCS.

doubleword -A contiguous sequence of bits or characters that comprises two computer words and is capable of being addressed as a unit. (A)

DPMI -DOS Protect Mode Interface.

drag -In SAA Common User Access, to use a pointing device to move an object; for example, clicking on a window border, and dragging it to make the window larger.

dragging -(1) In computer graphics, moving an object on the display screen as if it were attached to the pointer. (2) (D of C) In computer graphics, moving one or more segments on a display surface by translating. (I) (A)

drawing chain -See *segment chain* .

drop -To fix the position of an object that is being dragged, by releasing the select button of the pointing device.

drop -To fix the position of an object that is being dragged, by releasing the select button of the pointing device. See also *drag* .

DTL -Dialog tag language.

dual-boot function -A feature of the OS/2 operating system that allows the user to start DOS from within the operating system, or an OS/2 session from within DOS.

duplex -Pertaining to communication in which data can be sent and received at the same time. Synonymous with *full duplex* .

dynamic data exchange (DDE) -A message protocol used to communicate between applications that share data. The protocol uses shared memory as the means of exchanging data between applications.

dynamic data formatting -A formatting procedure that enables you to incorporate text, bit maps or metafiles in an IPF window at execution time.

dynamic link library -A collection of executable programming code and data that is bound to an application at load time or run time, rather than during linking. The programming code and data in a dynamic link library can be shared by several applications simultaneously.

dynamic linking -The process of resolving external references in a program module at load time or run time rather than during linking.

dynamic segments -Graphics segments drawn in exclusive-OR mix mode so that they can be moved from one screen position to another

without affecting the rest of the displayed picture.

dynamic storage -(1) A device that stores data in a manner that permits the data to move or vary with time such that the specified data is not always available for recovery. (A) (2) A storage in which the cells require repetitive application of control signals in order to retain stored data. Such repetitive application of the control signals is called a refresh operation. A dynamic storage may use static addressing or sensing circuits. (A) (3) See also *static storage* .

dynamic time slicing -Varies the size of the time slice depending on system load and paging activity.

dynamic-link module -A module that is linked at load time or run time.

Glossary - E

EBCDIC -Extended binary-coded decimal interchange code. A coded character set consisting of 8-bit coded characters (9 bits including parity check), used for information interchange among data processing systems, data communications systems, and associated equipment.

edge-triggered -Pertaining to an event semaphore that is posted then reset before a waiting thread gets a chance to run. The semaphore is considered to be posted for the rest of that thread's waiting period; the thread does not have to wait for the semaphore to be posted again.

EGA -Extended graphics adapter.

element -An entry in a graphics segment that comprises one or more graphics orders and that is addressed by the element pointer.

EMS -Expanded Memory Specification.

encapsulation -Hiding an object's implementation, that is, its private, internal data and methods. Private variables and methods are accessible only to the object that contains them.

entry field -In SAA Common User Access architecture, an area where a user types information. Its boundaries are usually indicated. See also *selection field* .

entry panel -A defined panel type containing one or more entry fields and protected information such as headings, prompts, and explanatory text.

entry-field control -The component of a user interface that provides the means by which the application receives data entered by the user in an entry field. When it has the input focus, the entry field displays a flashing pointer at the position where the next typed character will go.

environment segment -The list of environment variables and their values for a process.

environment strings -ASCII text strings that define the value of environment variables.

environment variables -Variables that describe the execution environment of a process. These variables are named by the operating system or by the application. Environment variables named by the operating system are PATH, DPATH, INCLUDE, INIT, LIB, PROMPT, and TEMP. The values of environment variables are defined by the user in the CONFIG.SYS file, or by using the SET command at the OS/2 command prompt.

error message -An indication that an error has been detected. (A)

event semaphore -A semaphore that enables a thread to signal a waiting thread or threads that an event has occurred or that a task has been completed. The waiting threads can then perform an action that is dependent on the completion of the signaled event.

exception -An abnormal condition such as an I/O error encountered in processing a data set or a file.

exclusive system semaphore -A system semaphore that can be modified only by threads within the same process.

executable file -(1) A file that contains programs or commands that perform operations or actions to be taken. (2) A collection of related data records that execute programs.

exit -To execute an instruction within a portion of a computer program in order to terminate the execution of that portion. Such portions of computer programs include loops, subroutines, modules, and so on. (T) Repeated exit requests return the user to the point from which all functions provided to the system are accessible. Contrast with *cancel* .

expanded memory specification (EMS) -Enables DOS applications to access memory above the 1MB real mode addressing limit.

extended attribute -An additional piece of information about a file object, such as its data format or category. It consists of a name and a value. A file object may have more than one extended attribute associated with it.

extended help -In SAA Common User Access architecture, a help action that provides information about the contents of the application window from which a user requested help. Contrast with *contextual help* .

extended-choice selection -A mode that allows the user to select more than one item from a window. Not all windows allow extended choice selection. Contrast with *multiple-choice selection* .

extent -Continuous space on a disk or diskette that is occupied by or reserved for a particular data set, data space, or file.

external link -In Information Presentation Facility, a link that connects external online document files.

Glossary - F

family-mode application -An application program that can run in the OS/2 environment and in the DOS environment; however, it cannot take advantage of many of the OS/2-mode facilities, such as multitasking, interprocess communication, and dynamic linking.

FAT -File allocation table.

FEA -Full extended attribute.

field-level help -Information specific to the field on which the cursor is positioned. This help function is "contextual" because it provides information about a specific item as it is currently used; the information is dependent upon the context within the work session.

FIFO -First-in-first-out. (A)

file -A named set of records stored or processed as a unit. (T)

file allocation table (FAT) -In IBM personal computers, a table used by the operating system to allocate space on a disk for a file, and to locate and chain together parts of the file that may be scattered on different sectors so that the file can be used in a random or sequential manner.

file attribute -Any of the attributes that describe the characteristics of a file.

File Manager -In the Presentation Manager, a program that displays directories and files, and allows various actions on them.

file specification -The full identifier for a file, which includes its drive designation, path, file name, and extension.

file system -The combination of software and hardware that supports storing information on a storage device.

file system driver (FSD) -A program that manages file I\O and controls the format of information on the storage media.

fillet -A curve that is tangential to the end points of two adjoining lines. See also *polyfillet* .

filtering -An application process that changes the order of data in a queue.

first-in-first-out (FIFO) -A queuing technique in which the next item to be retrieved is the item that has been in the queue for the longest time. (A)

flag -(1) An indicator or parameter that shows the setting of a switch. (2) A character that signals the occurrence of some condition, such as the end of a word. (A) (3) (D of C) A characteristic of a file or directory that enables it to be used in certain ways. See also *archive* flag, *hidden flag* , and *read-only flag* .

focus -See *input focus* .

folder -A container used to organize objects.

font -A particular size and style of typeface that contains definitions of character sets, marker sets, and pattern sets.

Font Editor -A utility program provided with the IBM Developers Toolkit that enables the design and creation of new fonts.

foreground program -(1) The program with which the user is currently interacting. Also known as *interactive program* . Contrast with

background program . (2) (D of C) In multiprogramming, a high-priority program.

frame -The part of a window that can contain several different visual elements specified by the application, but drawn and controlled by the Presentation Manager. The frame encloses the client area.

frame styles -Standard window layouts provided by the Presentation Manager.

FSD -File system driver.

full-duplex -Synonym for *duplex* .

full-screen application -An application that has complete control of the screen.

function -(1) In a programming language, a block, with or without formal parameters, whose execution is invoked by means of a call. (2) A set of related control statements that cause one or more programs to be performed.

function key -A key that causes a specified sequence of operations to be performed when it is pressed, for example, F1 and Alt-K.

function key area -The area at the bottom of a window that contains function key assignments such as F1=Help.

Glossary - G

GDT -Global Descriptor Table.

general protection fault -An exception condition that occurs when a process attempts to use storage or a module that has some level of protection assigned to it, such as I/O privilege level. See also */OPL code segment* .

Global Descriptor Table (GDT) -A table that defines code and data segments available to all tasks in an application.

global dynamic-link module -A dynamic-link module that can be shared by all processes in the system that refer to the module name.

global file-name character -Either a question mark (?) or an asterisk (*) used as a variable in a file name or file name extension when referring to a particular file or group of files.

glyph -A graphic symbol whose appearance conveys information.

GPI -Graphics programming interface.

graphic primitive -In computer graphics, a basic element, such as an arc or a line, that is not made up of smaller parts and that is used to create diagrams and pictures. See also *graphics segment* .

graphics -(1) A picture defined in terms of graphic primitives and graphics attributes. (2) (D of C) The making of charts and pictures. (3) Pertaining to charts, tables, and their creation. (4) See *computer graphics, coordinate graphics, fixed-image graphics, interactive graphics, passive graphics, raster graphics* .

graphics attributes -Attributes that apply to graphic primitives. Examples are color, line type, and shading-pattern definition. See also *segment attributes* .

graphics field -The clipping boundary that defines the visible part of the presentation-page contents.

graphics mode -One of several states of a display. The mode determines the resolution and color content of the screen.

graphics model space -The conceptual coordinate space in which a picture is constructed after any model transforms have been applied. Also known as *model space* .

Graphics programming interface -The formally defined programming language that is between an IBM graphics program and the user of the program.

graphics segment -A sequence of related graphic primitives and graphics attributes. See also *graphic primitive* .

graying -The indication that a choice on a pull-down is unavailable.

group -A collection of logically connected controls. For example, the buttons controlling paper size for a printer could be called a group. See also *program group* .

Glossary - H

handle -(1) An identifier that represents an object, such as a device or window, to the Presentation Interface. (2) (D of C) In the Advanced DOS and OS/2 operating systems, a binary value created by the system that identifies a drive, directory, and file so that the file can be found and opened.

hard error -An error condition on a network that requires either that the system be reconfigured or that the source of the error be removed before the system can resume reliable operation.

header -(1) System-defined control information that precedes user data. (2) The portion of a message that contains control information for the message, such as one or more destination fields, name of the originating station, input sequence number, character string indicating the type of message, and priority level for the message.

heading tags -A document element that enables information to be displayed in windows, and that controls entries in the contents window controls placement of push buttons in a window, and defines the shape and size of windows.

heap -An area of free storage available for dynamic allocation by an application. Its size varies according to the storage requirements of the application.

help function -(1) A function that provides information about a specific field, an application panel, or information about the help facility. (2) (D of C) One or more display images that describe how to use application software or how to do a system operation.

Help index -In SAA Common User Access architecture, a help action that provides an index of the help information available for an application.

help panel -A panel with information to assist users that is displayed in response to a help request from the user.

help window -A Common-User-Access-defined secondary window that displays information when the user requests help.

hidden file -An operating system file that is not displayed by a directory listing.

hide button -In the OS/2 operating system, a small, square button located in the right-hand corner of the title bar of a window that, when selected, removes from the screen all the windows associated with that window. Contrast with *maximize button* . See also *restore button* .

hierarchical inheritance -The relationship between parent and child classes. An object that is lower in the inheritance hierarchy than another object, inherits all the characteristics and behaviors of the objects above it in the hierarchy.

hierarchy -A tree of segments beginning with the root segment and proceeding downward to dependent segment types.

high-performance file system (HPFS) -In the OS/2 operating system, an installable file system that uses high-speed buffer storage, known as a cache, to provide fast access to large disk volumes. The file system also supports the coexistence of multiple, active file systems on a single personal computer, with the capability of multiple and different storage devices. File names used with the HPFS can have as many as 254 characters.

hit testing -The means of identifying which window is associated with which input device event.

hook -A point in a system-defined function where an application can supply additional code that the system processes as though it were part of the function.

hook chain -A sequence of hook procedures that are "chained" together so that each event is passed, in turn, to each procedure in the chain.

hot spot -The part of the pointer that must touch an object before it can be selected. This is usually the tip of the pointer. Contrast with *action point* .

HPFS -high-performance file system.

hypergraphic link -A connection between one piece of information and another through the use of graphics.

hypertext -A way of presenting information online with connections between one piece of information and another, called *hypertext links* . See also *hypertext link* .

hypertext link -A connection between one piece of information and another.

Glossary - I

I/O operation -An input operation to, or output operation from a device attached to a computer.

I-beam pointer -A pointer that indicates an area, such as an entry field in which text can be edited.

icon -In SAA Advanced Common User Access architecture, a graphical representation of an object, consisting of an image, image background, and a label. Icons can represent items (such as a document file) that the user wants to work on, and actions that the user wants to perform. In the Presentation Manager, icons are used for data objects, system actions, and minimized programs.

icon area -In the Presentation Manager, the area at the bottom of the screen that is normally used to display the icons for minimized programs.

Icon Editor -The Presentation Manager-provided tool for creating icons.

IDL -Interface Definition Language.

image font -A set of symbols, each of which is described in a rectangular array of pels. Some of the pels in the array are set to produce the image of one of the symbols. Contrast with *outline font* .

implied metaclass -Subclassing the metaclass of a parent class without a separate CSC for the resultant metaclass.

indirect manipulation -Interaction with an object through choices and controls.

information device context -A logical description of a data destination other than the screen (for example, a printer or plotter), but where no output will occur. Its purpose is to satisfy queries. See also *device context* .

information panel -A defined panel type characterized by a body containing only protected information.

Information Presentation Facility (IPF) -A facility provided by the OS/2 operating system, by which application developers can produce online documentation and context-sensitive online help panels for their applications.

inheritance -The derivation of new (child) classes from existing (parent) classes. The new class inherits all the data and methods of the parent class without having to redefine them.

input focus -(1) The area of a window where user interaction is possible using an input device, such as a mouse or the keyboard. (2) The position in the *active window* where a user's normal interaction with the keyboard will appear.

input router -An internal OS/2 process that removes messages from the system queue.

input/output control -A device-specific command that requests a function of a device driver.

installable file system (IFS) -A file system in which software is installed when the operating system is started.

instance -A single occurrence of an object class that has a particular behavior.

instruction pointer -In System/38, a pointer that provides addressability for a machine interface instruction in a program.

integer atom -An *atom* that represents a predefined system constant and carries no storage overhead. For example, names of window classes provided by Presentation Manager are expressed as integer atoms.

interactive graphics -Graphics that can be moved or manipulated by a user at a terminal.

interactive program -(1) A program that is running (active) and is ready to receive (or is receiving) input from a user. (2) A running program that can receive input from the keyboard or another input device. Compare with *active program* and contrast with *noninteractive program* .

Also known as a *foreground program* .

interchange file -A file containing data that can be sent from one Presentation Manager interface application to another.

Interface Definition Language (IDL) -Language-neutral interface specification for a SOM class.

interpreter -A program that translates and executes each instruction of a high-level programming language before it translates and executes.

interprocess communication (IPC) -In the OS/2 operating system, the exchange of information between processes or threads through semaphores, pipes, queues, and shared memory.

interval timer -(1) A timer that provides program interruptions on a program-controlled basis. (2) An electronic counter that counts intervals of time under program control.

IOCtl -Input/output control.

IOPL -Input/output privilege level.

IOPL code segment -An IOPL executable section of programming code that enables an application to directly manipulate hardware interrupts and ports without replacing the device driver. See also *privilege level* .

IPC -Interprocess communication.

IPF -Information Presentation Facility.

IPF compiler -A text compiler that interpret tags in a source file and converts the information into the specified format.

IPF tag language -A markup language that provides the instructions for displaying online information.

item -A data object that can be passed in a DDE transaction.

Glossary - J

journal -A special-purpose file that is used to record changes made in the system.

Glossary - K

Kanji -A graphic character set used in Japanese ideographic alphabets.

KBD$ -Character-device name reserved for the keyboard.

kernel -The part of an operating system that performs basic functions, such as allocating hardware resources.

kerning -The design of graphics characters so that their character boxes overlap. Used to space text proportionally.

keyboard accelerator -A keystroke that generates a command message for an application.

keyboard augmentation -A function that enables a user to press a keyboard key while pressing a mouse button.

keyboard focus -A temporary attribute of a window. The window that has a keyboard focus receives all keyboard input until the focus changes to a different window.

Keys help -In SAA Common User Access architecture, a help action that provides a listing of the application keys and their assigned functions.

Glossary - L

label -In a graphics segment, an identifier of one or more elements that is used when editing the segment.

LAN -local area network.

language support procedure -A function provided by the Presentation Manager Interface for applications that do not, or cannot (as in the case of COBOL and FORTRAN programs), provide their own dialog or window procedures.

lazy drag -See *pickup and drop* .

lazy drag set -See *pickup set* .

LDT -In the OS/2 operating system, Local Descriptor Table.

LIFO stack -A stack from which data is retrieved in last-in, first-out order.

linear address -A unique value that identifies the memory object.

linked list -Synonym for *chained list* .

list box -In SAA Advanced Common User Access architecture, a control that contains scrollable choices from which a user can select one choice.

Note: In CUA architecture, this is a programmer term. The end user term is selection list.

list button -A button labeled with an underlined down-arrow that presents a list of valid objects or choices that can be selected for that field.

list panel -A defined panel type that displays a list of items from which users can select one or more choices and then specify one or more actions to work on those choices.

load time -The point in time at which a program module is loaded into main storage for execution.

load-on-call -A function of a linkage editor that allows selected segments of the module to be disk resident while other segments are executing. Disk resident segments are loaded for execution and given control when any entry point that they contain is called.

local area network (LAN) -(1) A computer network located on a user's premises within a limited geographical area. Communication within a local area network is not subject to external regulations; however, communication across the LAN boundary may be subject to some form of regulation. (T)

Note: A LAN does not use store and forward techniques. (2) A network inwhich a set of devices are connected to one another for communication and that can be connected to a larger network.

Local Descriptor Table (LDT) -Defines code and data segments specific to a single task.

lock -A serialization mechanism by means of which a resource is restricted for use by the holder of the lock.

logical storage device -A device that the user can map to a physical (actual) device.

LPT1, LPT2, LPT3 -Character-device names reserved for parallel printers 1 through 3.

Glossary - M

main window -The window that is positioned relative to the *desktop window* .

manipulation button -The button on a pointing device a user presses to directly manipulate an object.

map -(1) A set of values having a defined correspondence with the quantities or values of another set. (I) (A) (2) To establish a set of values having a defined correspondence with the quantities or values of another set. (I)

marker box -In computer graphics, the boundary that defines, in world coordinates, the horizontal and vertical space occupied by a single marker from a marker set.

marker symbol -A symbol centered on a point. Graphs and charts can use marker symbols to indicate the plotted points.

marquee box -The rectangle that appears during a selection technique in which a user selects objects by drawing a box around them with a pointing device.

Master Help Index -In the OS/2 operating system, an alphabetic list of help topics related to using the operating system.

maximize -To enlarge a window to its largest possible size.

media window -The part of the physical device (display, printer, or plotter) on which a picture is presented.

memory block -Part memory within a heap.

memory device context -A logical description of a data destination that is a memory bit map. See also *device context* .

memory management -A feature of the operating system for allocating, sharing, and freeing main storage.

memory object -Logical unit of memory requested by an application, which forms the granular unit of memory manipulation from the application viewpoint.

menu -In SAA Advanced Common User Access architecture, an extension of the menu bar that displays a list of choices available for a selected choice in the menu bar. After a user selects a choice in menu bar, the corresponding menu appears. Additional pop-up windows can appear from menu choices.

menu bar -In SAA Advanced Common User Access architecture, the area near the top of a window, below the title bar and above the rest of the window, that contains choices that provide access to other menus.

menu button -The button on a pointing device that a user presses to view a pop-up menu associated with an object.

message -(1) In the Presentation Manager, a packet of data used for communication between the Presentation Manager interface and Presentation Manager applications (2) In a user interface, information not requested by users but presented to users by the computer in response to a user action or internal process.

message box -(1) A dialog window predefined by the system and used as a simple interface for applications, without the necessity of creating dialog-template resources or dialog procedures. (2) (D of C) In SAA Advanced Common User Access architecture, a type of window that shows messages to users. See also *dialog box, primary window, secondary window* .

message filter -The means of selecting which messages from a specific window will be handled by the application.

message queue -A sequenced collection of messages to be read by the application.

message stream mode -A method of operation in which data is treated as a stream of messages. Contrast with *byte stream* .

metacharacter -See *global file-name character* .

metaclass -The conjunction of an object and its class information; that is, the information pertaining to the class as a whole, rather than to a single instance of the class. Each class is itself an object, which is an instance of the metaclass.

metafile -A file containing a series of attributes that set color, shape and size, usually of a picture or a drawing. Using a program that can interpret these attributes, a user can view the assembled image.

metafile device context -A logical description of a data destination that is a metafile, which is used for graphics interchange. See also *device context* .

metalanguage -A language used to specify another language. For example, data types can be described using a metalanguage so as to make the descriptions independent of any one computer language.

method -A function that defines a behavior for a class or object.

method override -The replacement, by a child class, of the implementation of a method inherited from a parent and an ancestor class.

mickey -A unit of measurement for physical mouse motion whose value depends on the mouse device driver currently loaded.

micro presentation space -A graphics presentation space in which a restricted set of the GPI function calls is available.

minimize -To remove from the screen all windows associated with an application and replace them with an icon that represents the application.

mix -An attribute that determines how the foreground of a graphic primitive is combined with the existing color of graphics output. Also known as *foreground mix* . Contrast with *background mix* .

mixed character string -A string containing a mixture of one-byte and *Kanji* or Hangeul (two-byte) characters.

mnemonic -(1) A method of selecting an item on a pull-down by means of typing the highlighted letter in the menu item. (2) (D of C) In SAA Advanced Common User Access architecture, usually a single character, within the text of a choice, identified by an underscore beneath the character. If all characters in a choice already serve as mnemonics for other choices, another character, placed in parentheses immediately following the choice, can be used. When a user types the mnemonic for a choice, the choice is either selected or the cursor is moved to that choice.

modal dialog box -In SAA Advanced Common User Access architecture, a type of movable window, fixed in size, that requires a user to enter information before continuing to work in the application window from which it was displayed. Contrast with *modeless dialog box* . Also known as a *serial dialog box* . Contrast with *parallel dialog box* .

Note: In CUA architecture, this is a programmer term. The end user term is pop-up window.

model space -See *graphics model space* .

modeless dialog box -In SAA Advanced Common User Access architecture, a type of movable window, fixed in size, that allows users to continue their dialog with the application without entering information in the dialog box. Also known as a *parallel dialog box* . Contrast with *modal dialog box* .

Note: In CUA architecture, this is a programmer term. The end user term is pop-up window.

module definition file -A file that describes the code segments within a load module. For example, it indicates whether a code segment is loadable before module execution begins (preload), or loadable only when referred to at run time (load-on-call).

mouse -In SAA usage, a device that a user moves on a flat surface to position a pointer on the screen. It allows a user to select a choice o function to be performed or to perform operations on the screen, such as dragging or drawing lines from one position to another.

MOUSE$ -Character-device name reserved for a mouse.

multiple-choice selection -In SAA Basic Common User Access architecture, a type of field from which a user can select one or more choices or select none. See also *check box* . Contrast with *extended-choice selection* .

multiple-line entry field -In SAA Advanced Common User Access architecture, a control into which a user types more than one line of information. See also *single-line entry field* .

multitasking -The concurrent processing of applications or parts of applications. A running application and its data are protected from other concurrently running applications.

mutex semaphore -(Mutual exclusion semaphore). A semaphore that enables threads to serialize their access to resources. Only the thread that currently owns the mutex semaphore can gain access to the resource, thus preventing one thread from interrupting operations being performed by another.

muxwait semaphore -(Multiple wait semaphore). A semaphore that enables a thread to wait either for multiple event semaphores to be posted or for multiple mutex semaphores to be released. Alternatively, a muxwait semaphore can be set to enable a thread to wait for any ONE of the event or mutex semaphores in the muxwait semaphore's list to be posted or released.

Glossary - N

named pipe -A named buffer that provides client-to-server, server-to-client, or full duplex communication between unrelated processes. Contrast with *unnamed pipe* .

national language support (NLS) -The modification or conversion of a United States English product to conform to the requirements of another language or country. This can include the enabling or retrofitting of a product and the translation of nomenclature, MRI, or documentation of a product.

nested list -A list that is contained within another list.

NLS -national language support.

non-8.3 file-name format -A file-naming convention in which file names can consist of up to 255 characters. See also *8.3 file-name format* .

noncritical extended attribute -An extended attribute that is not necessary for the function of an application.

nondestructive read -Reading that does not erase the data in the source location. (T)

noninteractive program -A running program that cannot receive input from the keyboard or other input device. Compare with *active program* , and contrast with *interactive program* .

nonretained graphics -Graphic primitives that are not remembered by the Presentation Manager interface when they have been drawn. Contrast with *retained graphics* .

null character (NUL) -(1) Character-device name reserved for a nonexistent (dummy) device. (2) (D of C) A control character that is used to

accomplish media-fill or time-fill and that may be inserted into or removed from a sequence of characters without affecting the meaning of the sequence; however, the control of equipment or the format may be affected by this character. (I) (A)

null-terminated string -A string of (n+1) characters where the (n+1)th character is the 'null' character (0x00) Also known as 'zero-terminated' string and 'ASCIIZ' string.

Glossary - O

object -A set of data and actions that can be performed on that data.

Object Interface Definition Language (OIDL) -Specification language used in SOM Version 1 for defining classes. Replaced by Interface Definition Language (IDL).

object window -A window that does not have a parent but which might have child windows. An object window cannot be presented on a device.

OIDL -Object Interface Definition Language.

open -To start working with a file, directory, or other object.

ordered list -Vertical arrangements of items, with each item in the list preceded by a number or letter.

outline font -A set of symbols, each of which is created as a series of lines and curves. Synonymous with *vector font* . Contrast with *image font* .

output area -An area of storage reserved for output. (A)

owner window -A window into which specific events that occur in another (owned) window are reported.

ownership -The determination of how windows communicate using messages.

owning process -The process that owns the resources that might be shared with other processes.

Glossary - P

page -(1) A 4KB segment of contiguous physical memory. (2) (D of C) A defined unit of space on a storage medium.

page viewport -A boundary in device coordinates that defines the area of the output device in which graphics are to be displayed. The presentation-page contents are transformed automatically to the page viewport in device space.

paint -(1) The action of drawing or redrawing the contents of a window. (2) In computer graphics, to shade an area of a display image; for example, with crosshatching or color.

panel -In SAA Basic Common User Access architecture, a particular arrangement of information that is presented in a window or pop-up. If some of the information is not visible, a user can scroll through the information.

panel area -An area within a panel that contains related information. The three major Common User Access-defined panel areas are the action bar, the function key area, and the panel body.

panel area separator -In SAA Basic Common User Access architecture, a solid, dashed, or blank line that provides a visual distinction between two adjacent areas of a panel.

panel body -The portion of a panel not occupied by the action bar, function key area, title or scroll bars. The panel body can contain protected information, selection fields, and entry fields. The layout and content of the panel body determine the panel type.

panel body area -See *client area* .

panel definition -A description of the contents and characteristics of a panel. A panel definition is the application developer's mechanism for

predefining the format to be presented to users in a window.

panel ID -In SAA Basic Common User Access architecture, a panel identifier, located in the upper-left corner of a panel. A user can choose whether to display the panel ID.

panel title -In SAA Basic Common User Access architecture, a particular arrangement of information that is presented in a window or pop-up. If some of the information is not visible, a user can scroll through the information.

paper size -The size of paper, defined in either standard U.S. or European names (for example, A, B, A4), and measured in inches or millimeters respectively.

parallel dialog box -See *modeless dialog box* .

parameter list -A list of values that provides a means of associating addressability of data defined in a called program with data in the calling program. It contains parameter names and the order in which they are to be associated in the calling and called program.

parent process -In the OS/2 operating system, a process that creates other processes. Contrast with *child process* .

parent window -In the OS/2 operating system, a window that creates a child window. The child window is drawn within the parent window. If the parent window is moved, resized, or destroyed, the child window also will be moved, resized, or destroyed. However, the child window can be moved and resized independently from the parent window, within the boundaries of the parent window. Contrast with *child window* .

partition -(1) A fixed-size division of storage. (2) On an IBM personal computer fixed disk, one of four possible storage areas of variable size; one may be accessed by DOS, and each of the others may be assigned to another operating system.

Paste -A choice in the Edit pull-down that a user selects to move the contents of the clipboard into a preselected location. See also *Copy* and *Cut* .

path -The route used to locate files; the storage location of a file. A fully qualified path lists the drive identifier, directory name, subdirectory name (if any), and file name with the associated extension.

PDD -Physical device driver.

peeking -An action taken by any thread in the process that owns the queue to examine queue elements without removing them.

pel -(1) The smallest area of a display screen capable of being addressed and switched between visible and invisible states. Synonym for *display point* , *pixel* , and *picture element* . (2) (D of C) Picture element.

persistent object -An object whose instance data and state are preserved between system shutdown and system startup.

physical device driver (PDD) -A system interface that handles hardware interrupts and supports a set of input and output functions.

pick -To select part of a displayed object using the pointer.

pickup -To add an object or set of objects to the pickup set.

pickup and drop -A drag operation that does not require the direct manipulation button to be pressed for the duration of the drag.

pickup set -The set of objects that have been picked up as part of a pickup and drop operation.

picture chain -See *segment chain* .

picture element -(1) Synonym for *pel* . (2) (D of C) In computer graphics, the smallest element of a display surface that can be independently assigned color and intensity. (T) . (3) The area of the finest detail that can be reproduced effectively on the recording medium.

PID -Process identification.

pipe -(1) A named or unnamed buffer used to pass data between processes. A process reads from or writes to a pipe as if the pipe were a standard-input or standard-output file. See also *named pipe* and *unnamed pipe* . (2) (D of C) To direct data so that the output from one process becomes the input to another process. The standard output of one command can be connected to the standard input of another with the pipe operator (|).

pixel -(1) Synonym for *pel* . (2) (D of C) Picture element.

plotter -An output unit that directly produces a hardcopy record of data on a removable medium, in the form of a two-dimensional graphic representation. (T)

PM -Presentation Manager.

pointer -(1) The symbol displayed on the screen that is moved by a pointing device, such as a *mouse* . The pointer is used to point at items that users can select. Contrast with *cursor* . (2) A data element that indicates the location of another data element. (T)

POINTER$ -Character-device name reserved for a pointer device (mouse screen support).

pointing device -In SAA Advanced Common User Access architecture, an instrument, such as a mouse, trackball, or joystick, used to move a pointer on the screen.

pointings -Pairs of x-y coordinates produced by an operator defining positions on a screen with a pointing device, such as a *mouse* .

polyfillet -A curve based on a sequence of lines. The curve is tangential to the end points of the first and last lines, and tangential also to the midpoints of all other lines. See also *fillet* .

polygon -One or more closed figures that can be drawn filled, outlined, or filled and outlined.

polyline -A sequence of adjoining lines.

polymorphism -The ability to have different implementations of the same method for two or more classes of objects.

pop -To retrieve an item from a last-in-first-out stack of items. Contrast with *push* .

pop-up menu -A menu that lists the actions that a user can perform on an object. The contents of the pop-up menu can vary depending on the context, or state, of the object.

pop-up window -(1) A window that appears on top of another window in a dialog. Each pop-up window must be completed before returning to the underlying window. (2) (D of C) In SAA Advanced Common User Access architecture, a movable window, fixed in size, in which a user provides information required by an application so that it can continue to process a user request.

presentation drivers -Special purpose I/O routines that handle field device-independent I/O requests from the PM and its applications.

Presentation Manager (PM) -The interface of the OS/2 operating system that presents, in windows a graphics-based interface to applications and files installed and running under the OS/2 operating system.

presentation page -The coordinate space in which a picture is assembled for display.

presentation space (PS) -(1) Contains the device-independent definition of a picture. (2) (D of C) The display space on a display device.

primary window -In SAA Common User Access architecture, the window in which the main interaction between the user and the application takes place. In a multiprogramming environment, each application starts in its own primary window. The primary window remains for the duration of the application, although the panel displayed will change as the user's dialog moves forward. See also *secondary window* .

primitive -In computer graphics, one of several simple functions for drawing on the screen, including, for example, the rectangle, line, ellipse, polygon, and so on.

primitive attribute -A specifiable characteristic of a graphic primitive. See *graphics attributes* .

print job -The result of sending a document or picture to be printed.

Print Manager -In the Presentation Manager, the part of the spooler that manages the spooling process. It also allows users to view print queues and to manipulate print jobs.

privilege level -A protection level imposed by the hardware architecture of the IBM personal computer. There are four privilege levels (number 0 through 3). Only certain types of programs are allowed to execute at each privilege level. See also *IOPL code segment* .

procedure call -In programming languages, a language construct for invoking execution of a procedure.

process -An instance of an executing application and the resources it is using.

program -A sequence of instructions that a computer can interpret and execute.

program details -Information about a program that is specified in the *Program Manager* window and is used when the program is started.

program group -In the Presentation Manager, several programs that can be acted upon as a single entity.

program name -The full file specification of a program. Contrast with *program title* .

program title -The name of a program as it is listed in the *Program Manager* window. Contrast with *program name* .

prompt -A displayed symbol or message that requests input from the user or gives operational information; for example, on the display screen of an IBM personal computer, the DOS A> prompt. The user must respond to the prompt in order to proceed.

protect mode -A method of program operation that limits or prevents access to certain instructions or areas of storage. Contrast with *real mode* .

protocol -A set of semantic and syntactic rules that determines the behavior of functional units in achieving communication. (I)

pseudocode -An artificial language used to describe computer program algorithms without using the syntax of any particular programming language. (A)

pull-down -(1) An *action bar* extension that displays a list of choices available for a selected action bar choice. After users select an action bar choice, the pull-down appears with the list of choices. Additional *pop-up windows* may appear from pull-down choices to further extend the actions available to users. (2) (D of C) In SAA Common User Access architecture, pertaining to a choice in an action bar pull-down.

push -To add an item to a last-in-first-out stack of items. Contrast with *pop* .

push button -In SAA Advanced Common User Access architecture, a rectangle with text inside. Push buttons are used in windows for actions that occur immediately when the push button is selected.

putback -To remove an object or set of objects from the lazy drag set. This has the effect of undoing the pickup operation for those objects

putdown -To drop the objects in the lazy drag set on the target object.

Glossary - Q

queue -(1) A linked list of elements waiting to be processed in FIFO order. For example, a queue may be a list of print jobs waiting to be printed. (2) (D of C) A line or list of items waiting to be processed; for example, work to be performed or messages to be displayed.

queued device context -A logical description of a data destination (for example, a printer or plotter) where the output is to go through the spooler. See also *device context* .

Glossary - R

radio button -(1) A control window, shaped like a round button on the screen, that can be in a checked or unchecked state. It is used to select a single item from a list. Contrast with *check box* . (2) In SAA Advanced Common User Access architecture, a circle with text beside it. Radio buttons are combined to show a user a fixed set of choices from which only one can be selected. The circle is partially filled when a choice is selected.

RAS -Reliability, availability, and serviceability.

raster -(1) In computer graphics, a predetermined pattern of lines that provides uniform coverage of a display space. (T) (2) The coordinate grid that divides the display area of a display device. (A)

read-only file -A file that can be read from but not written to.

real mode -A method of program operation that does not limit or prevent access to any instructions or areas of storage. The operating system loads the entire program into storage and gives the program access to all system resources. Contrast with *protect mode* .

realize -To cause the system to ensure, wherever possible, that the physical color table of a device is set to the closest possible match in the logical color table.

recursive routine -A routine that can call itself, or be called by another routine that was called by the recursive routine.

reentrant -The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks.

reference phrase -(1) A word or phrase that is emphasized in a device-dependent manner to inform the user that additional information for the word or phrase is available. (2) (D of C) In hypertext, text that is highlighted and preceded by a single-character input field used to signify the existence of a hypertext link.

reference phrase help -In SAA Common User Access architecture, highlighted words or phrases within help information that a user selects to get additional information.

refresh -To update a window, with changed information, to its current status.

region -A clipping boundary in device space.

register -A part of internal storage having a specified storage capacity and usually intended for a specific purpose. (T)

remote file system -A file-system driver that gains access to a remote system without a block device driver.

resource -The means of providing extra information used in the definition of a window. A resource can contain definitions of fonts, templates, accelerators, and mnemonics; the definitions are held in a resource file.

resource file -A file containing information used in the definition of a window. Definitions can be of fonts, templates, accelerators, and mnemonics.

restore -To return a window to its original size or position following a sizing or moving action.

retained graphics -Graphic primitives that are remembered by the Presentation Manager interface after they have been drawn. Contrast with *nonretained graphics* .

return code -(1) A value returned to a program to indicate the results of an operation requested by that program. (2) A code used to influence the execution of succeeding instructions.(A)

reverse video -(1) A form of highlighting a character, field, or cursor by reversing the color of the character, field, or cursor with its background; for example, changing a red character on a black background to a black character on a red background. (2) In SAA Basic Common User Access architecture, a screen emphasis feature that interchanges the foreground and background colors of an item.

REXX Language -Restructured Extended Executor. A procedural language that provides batch language functions along with structured programming constructs such as loops; conditional testing and subroutines.

RGB -(1) Color coding in which the brightness of the additive primary colors of light, red, green, and blue, are specified as three distinct values of white light. (2) Pertaining to a color display that accepts signals representing red, green, and blue.

roman -Relating to a type style with upright characters.

root segment -In a hierarchical database, the highest segment in the tree structure.

round-robin scheduling -A process that allows each thread to run for a specified amount of time.

run time -(1) Any instant at which the execution of a particular computer program takes place. (T) (2) The amount of time needed for the execution of a particular computer program. (T) (3) The time during which an instruction in an instruction register is decoded and performed. Synonym for *execution time* .

Glossary - S

SAA -Systems Application Architecture.

SBCS -Single-byte character set.

scheduler -A computer program designed to perform functions such as scheduling, initiation, and termination of jobs.

screen -In SAA Basic Common User Access architecture, the physical surface of a display device upon which information is shown to a user.

screen device context -A logical description of a data destination that is a particular window on the screen. See also *device context* .

SCREEN$ -Character-device name reserved for the display screen.

scroll bar -In SAA Advanced Common User Access architecture, a part of a window, associated with a scrollable area, that a user interacts with to see information that is not currently allows visible.

scrollable entry field -An entry field larger than the visible field.

scrollable selection field -A selection field that contains more choices than are visible.

scrolling -Moving a display image vertically or horizontally in a manner such that new data appears at one edge, as existing data disappears at the opposite edge.

secondary window -A window that contains information that is dependent on information in a primary window and is used to supplement the interaction in the primary window.

sector -On disk or diskette storage, an addressable subdivision of a track used to record one block of a program or data.

segment -See *graphics segment* .

segment attributes -Attributes that apply to the segment as an entity, as opposed to the individual primitives within the segment. For example, the visibility or detectability of a segment.

segment chain -All segments in a graphics presentation space that are defined with the 'chained' attribute. Synonym for *picture chain* .

segment priority -The order in which segments are drawn.

segment store -An area in a normal graphics presentation space where retained graphics segments are stored.

select -To mark or choose an item. Note that *select* means to mark or type in a choice on the screen; *enter* means to send all selected choices to the computer for processing.

select button -The button on a pointing device, such as a mouse, that is pressed to select a menu choice. Also known as button 1.

selection cursor -In SAA Advanced Common User Access architecture, a visual indication that a user has selected a choice. It is represented by outlining the choice with a dotted box. See also *text cursor* .

selection field -(1) In SAA Advanced Common User Access architecture, a set of related choices. See also *entry field* . (2) In SAA Basic Common User Access architecture, an area of a panel that cannot be scrolled and contains a fixed number of choices.

semantics -The relationships between symbols and their meanings.

semaphore -An object used by applications for signalling purposes and for controlling access to serially reusable resources.

separator -In SAA Advanced Common User Access architecture, a line or color boundary that provides a visual distinction between two adjacent areas.

serial dialog box -See *modal dialog box* .

serialization -The consecutive ordering of items.

serialize -To ensure that one or more events occur in a specified sequence.

serially reusable resource (SRR) -A logical resource or object that can be accessed by only one task at a time.

session -(1) A routing mechanism for user interaction via the console; a complete environment that determines how an application runs and how users interact with the application. OS/2 can manage more than one session at a time, and more than one process can run in a session. Each session has its own set of environment variables that determine where OS/2 looks for dynamic-link libraries and other important files. (2) (D of C) In the OS/2 operating system, one instance of a started program or command prompt. Each session is separate from all other sessions that might be running on the computer. The operating system is responsible for coordinating the resources that each session uses, such as computer memory, allocation of processor time, and windows on the screen.

Settings Notebook -A control window that is used to display the settings for an object and to enable the user to change them.

shadow -An object that refers to another object. A shadow is not a copy of another object, but is another representation of the object.

shadow box -The area on the screen that follows mouse movements and shows what shape the window will take if the mouse button is released.

shared data -Data that is used by two or more programs.

shared memory -In the OS/2 operating system, a segment that can be used by more than one program.

shear -In computer graphics, the forward or backward slant of a graphics symbol or string of such symbols relative to a line perpendicular to the baseline of the symbol.

shell -(1) A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens, and communicate them to the operating system. (2) Software that allows a kernel program to run under different operating-system environments.

shutdown -The process of ending operation of a system or a subsystem, following a defined procedure.

sibling processes -Child processes that have the same parent process.

sibling windows -Child windows that have the same parent window.

simple list -A list of like values; for example, a list of user names. Contrast with *mixed list* .

single-byte character set (SBCS) -A character set in which each character is represented by a one-byte code. Contrast with *double-byte character set* .

slider box -In SAA Advanced Common User Access architecture: a part of the scroll bar that shows the position and size of the visible information in a window relative to the total amount of information available. Also known as *thumb mark* .

SOM -System Object Model.

source file -A file that contains source statements for items such as high-level language programs and data description specifications.

source statement -A statement written in a programming language.

specific dynamic-link module -A dynamic-link module created for the exclusive use of an application.

spin button -In SAA Advanced Common User Access architecture, a type of entry field that shows a scrollable ring of choices from which a user can select a choice. After the last choice is displayed, the first choice is displayed again. A user can also type a choice from the scrollable ring into the entry field without interacting with the spin button.

spline -A sequence of one or more Bezier curves.

spooler -A program that intercepts the data going to printer devices and writes it to disk. The data is printed or plotted when it is complete and the required device is available. The spooler prevents output from different sources from being intermixed.

stack -A list constructed and maintained so that the next data element to be retrieved is the most recently stored. This method is characterized as last-in-first-out (LIFO).

standard window -A collection of window elements that form a panel. The standard window can include one or more of the following window elements: sizing borders, system menu icon, title bar, maximize/minimize/restore icons, action bar and pull-downs, scroll bars, and client area.

static control -The means by which the application presents descriptive information (for example, headings and descriptors) to the user. The user cannot change this information.

static storage -(1) A read/write storage unit in which data is retained in the absence of control signals. (A) Static storage may use dynamic addressing or sensing circuits. (2) Storage other than *dynamic storage* . (A)

style -See *window style* .

subclass -A class that is a child of another class. See also Inheritance.

subdirectory -In an IBM personal computer, a file referred to in a root directory that contains the names of other files stored on the diskette or fixed disk.

swapping -(1) A process that interchanges the contents of an area of real storage with the contents of an area in auxiliary storage. (I) (A) (2) In a system with virtual storage, a paging technique that writes the active pages of a job to auxiliary storage and reads pages of another job from auxiliary storage into real storage. (3) The process of temporarily removing an active job from main storage, saving it on disk, and processing another job in the area of main storage formerly occupied by the first job.

switch -(1) In SAA usage, to move the cursor from one point of interest to another; for example, to move from one screen or window to another or from a place within a displayed image to another place on the same displayed image. (2) In a computer program, a conditional instruction and an indicator to be interrogated by that instruction. (3) A device or programming technique for making a selection, for example, a toggle, a conditional jump.

switch list -See *Task List* .

symbolic identifier -A text string that equates to an integer value in an include file, which is used to identify a programming object.

symbols -In Information Presentation Facility, a document element used to produce characters that cannot be entered from the keyboard.

synchronous -Pertaining to two or more processes that depend upon the occurrence of specific events such as common timing signals. (T) See also *asynchronous* .

System Menu -In the Presentation Manager, the pull-down in the top left corner of a window that allows it to be moved and sized with the keyboard.

System Object Model (SOM) -A mechanism for language-neutral, object-oriented programming in the OS/2 environment.

system queue -The master queue for all pointer device or keyboard events.

system-defined messages -Messages that control the operations of applications and provides input an other information for applications to process.

Systems Application Architecture (SAA) -A set of IBM software interfaces, conventions, and protocols that provide a framework for designing and developing applications that are consistent across systems.

Glossary - T

table tags -In Information Presentation Facility, a document element that formats text in an arrangement of rows and columns.

tag -(1) One or more characters attached to a set of data that contain information about the set, including its identification. (I) (A) (2) In Generalized Markup Language markup, a name for a type of document or document element that is entered in the source document to identify it.

target object -An object to which the user is transferring information.

Task List -In the Presentation Manager, the list of programs that are active. The list can be used to switch to a program and to stop programs.

terminate-and-stay-resident (TSR) -Pertaining to an application that modifies an operating system interrupt vector to point to its own location (known as hooking an interrupt).

text -Characters or symbols.

text cursor -A symbol displayed in an entry field that indicates where typed input will appear.

text window -Also known as the VIO window.

text-windowed application -The environment in which the operating system performs advanced-video input and output operations.

thread -A unit of execution within a process. It uses the resources of the process.

thumb mark -The portion of the scroll bar that describes the range and properties of the data that is currently visible in a window. Also known as a *slider box* .

thunk -Term used to describe the process of address conversion, stack and structure realignment, etc., necessary when passing control between 16-bit and 32-bit modules.

tilde -A mark used to denote the character that is to be used as a mnemonic when selecting text items within a menu.

time slice -(1) An interval of time on the processing unit allocated for use in performing a task. After the interval has expired, processing-unit time is allocated to another task, so a task cannot monopolize processing-unit time beyond a fixed limit. (2) In systems with time sharing, a segment of time allocated to a terminal job.

time-critical process -A process that must be performed within a specified time after an event has occurred.

timer -A facility provided under the Presentation Manager, whereby Presentation Manager will dispatch a message of class WM_TIMER to a particular window at specified intervals. This capability may be used by an application to perform a specific processing task at predetermined intervals, without the necessity for the application to explicitly keep track of the passage of time.

timer tick -See *clock tick* .

title bar -In SAA Advanced Common User Access architecture, the area at the top of each window that contains the window title and system menu icon. When appropriate, it also contains the minimize, maximize, and restore icons. Contrast with *panel title* .

TLB -Translation lookaside buffer.

transaction -An exchange between a workstation and another device that accomplishes a particular action or result.

transform -(1) The action of modifying a picture by scaling, shearing, reflecting, rotating, or translating. (2) The object that performs or defines such a modification; also referred to as a *transformation* .

Translation lookaside buffer (TLB) -A hardware-based address caching mechanism for paging information.

Tree -In the Presentation Manager, the window in the *File Manager* that shows the organization of drives and directories.

truncate -(1) To terminate a computational process in accordance with some rule (A) (2) To remove the beginning or ending elements of a string. (3) To drop data that cannot be printed or displayed in the line width specified or available. (4) To shorten a field or statement to a specified length.

TSR -Terminate-and-stay-resident.

unnamed pipe -A circular buffer, created in memory, used by related processes to communicate with one another. Contrast with *named pipe* .

unordered list -In Information Presentation Facility, a vertical arrangement of items in a list, with each item in the list preceded by a special character or bullet.

update region -A system-provided area of dynamic storage containing one or more (not necessarily contiguous) rectangular areas of a window that are visually invalid or incorrect, and therefore are in need of repainting.

user interface -Hardware, software, or both that allows a user to interact with and perform operations on a system, program, or device.

User Shell -A component of OS/2 that uses a graphics-based, windowed interface to allow the user to manage applications and files installed and running under OS/2.

utility program -(1) A computer program in general support of computer processes; for example, a diagnostic program, a trace program, a sort program. (T) (2) A program designed to perform an everyday task such as copying data from one storage device to another. (A)

Glossary - U

There are no glossary terms for this starting letter.

Glossary - V

value set control -A visual component that enables a user to select one choice from a group of mutually exclusive choices.

vector font -A set of symbols, each of which is created as a series of lines and curves. Synonymous with *outline font* . Contrast with *image font* .

VGA -Video graphics array.

view -A way of looking at an object's information.

viewing pipeline -The series of transformations applied to a graphic object to map the object to the device on which it is to be presented.

viewing window -A clipping boundary that defines the visible part of model space.

VIO -Video Input/Output.

virtual memory (VM) -Synonymous with *virtual storage* .

virtual storage -(1) The storage space that may be regarded as addressable main storage by the user of a computer system in which virtual addresses are mapped into real addresses. The size of virtual storage is limited by the addressing scheme of the computer system and by the amount of auxiliary storage available, not by the actual number of main storage locations. (I) (A) (2) Addressable space that is apparent to the user as the processor storage space, from which the instructions and the data are mapped into the processor storage locations. (3) Synonymous with *virtual memory* .

visible region -A window's presentation space, clipped to the boundary of the window and the boundaries of any overlying window.

volume -(1) A file-system driver that uses a block device driver for input and output operations to a local or remote device. (I) (2) A portion of data, together with its data carrier, that can be handled conveniently as a unit.

Glossary - W

wildcard character -Synonymous with *global file-name character* .

window -(1) A portion of a display surface in which display images pertaining to a particular application can be presented. Different applications can be displayed simultaneously in different windows. (A) (2) An area of the screen with visible boundaries within which information is displayed. A window can be smaller than or the same size as the screen. Windows can appear to overlap on the screen.

window class -The grouping of windows whose processing needs conform to the services provided by one window procedure.

window coordinates -A set of coordinates by which a window position or size is defined; measured in device units, or *pels* .

window handle -Unique identifier of a window, generated by Presentation Manager when the window is created, and used by applications to direct messages to the window.

window procedure -Code that is activated in response to a message. The procedure controls the appearance and behavior of its associated windows.

window rectangle -The means by which the size and position of a window is described in relation to the desktop window.

window resource -A read-only data segment stored in the .EXE file of an application o the .DLL file of a dynamic link library.

window style -The set of properties that influence how events related to a particular window will be processed.

window title -In SAA Advanced Common User Access architecture, the area in the title bar that contains the name of the application and the OS/2 operating system file name, if applicable.

Workplace Shell -The OS/2 object-oriented, graphical user interface.

workstation -(1) A display screen together with attachments such as a keyboard, a local copy device, or a tablet. (2) (D of C) One or more programmable or nonprogrammable devices that allow a user to do work.

world coordinates -A device-independent Cartesian coordinate system used by the application program for specifying graphical input and output. (I) (A)

world-coordinate space -Coordinate space in which graphics are defined before transformations are applied.

WYSIWYG -What-You-See-Is-What-You-Get. A capability of a text editor to continually display pages exactly as they will be printed.

Glossary - X

There are no glossary terms for this starting letter.

Glossary - Y

There are no glossary terms for this starting letter.

Glossary - Z

z-order -The order in which sibling windows are presented. The topmost sibling window obscures any portion of the siblings that it overlaps; the same effect occurs down through the order of lower sibling windows.

zooming -The progressive scaling of an entire display image in order to give the visual impression of movement of all or part of a display group toward or away from an observer. (I) (A)

8.3 file-name format -A file-naming convention in which file names are limited to eight characters before and three characters after a single dot. Usually pronounced "eight-dot-three." See also *non-8.3 file-name format* .

IBM Trademark

Trademark of the IBM Corporation.

Trademarks

Trademark of AT&T, Inc.

Trademarks

Trademark of the Adobe Systems Inc.

Trademarks

Trademark of the Microsoft Corporation.

Trademarks

Trademark of Linotype.

Trademarks

Trademark of Monotype Corporation, Limited.

Trademarks

Trademark of Intel Coporation.
